

AN INTERACTIVE HYBRID COMPUTER SYSTEM
FOR TIME DOMAIN AUDIO SYNTHESIS

James William Mizerski

Library
Naval Postgraduate School
Monterey, California 93940

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

AN INTERACTIVE HYBRID COMPUTER SYSTEM
FOR TIME DOMAIN AUDIO SYNTHESIS

by

James William Mizerski

Thesis Advisor:

V.M. Powers

March 1973

T152.69

Approved for public release; distribution unlimited.

An Interactive Hybrid Computer System
for Time Domain Audio Synthesis

by

James William Mizerski
Lieutenant, United States Naval Reserve
B.S., Loyola University of Los Angeles, 1967

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1973

Thesis
MG54
C-1

ABSTRACT

A time domain oriented technique using periodic functions composed of first order segments was employed in the development of a real-time interactive computer audio synthesis system. A general purpose analog computer was programmed to create non-monotonic function generators which are used to generate sequences of periodic functions that can be frequency and amplitude modulated. The system is appropriate for experiments in psychoacoustics. Exploration of the relationships between the physical and the perceptual significance of synthesis parameters of the technique is aided by real-time frequency analysis programs and an audio output capability. The system includes an interactive graphics terminal for conversational I/O and conventional peripheral devices for creating permanent records. In preliminary testing, the technique and system displayed promising possibilities. Modifications to explore specific areas of perception in greater depth were considered.

TABLE OF CONTENTS

I.	INTRODUCTION -----	7
A.	BACKGROUND -----	7
B.	PURPOSE AND APPROACH -----	16
C.	PROBLEMS AND SOLUTIONS -----	18
II.	SYSTEM DESCRIPTION -----	23
A.	GENERAL SPECIFICATIONS -----	23
1.	User Oriented -----	23
2.	Permanent Records -----	24
3.	Simple Time Domain Models -----	24
4.	Frequency Analysis -----	25
5.	Signal Output -----	28
B.	SYSTEM COMPONENTS -----	28
1.	Major Hardware Components -----	28
a.	XDS 9300 -----	28
b.	COMCOR CI-5000 -----	28
c.	ADAGE AGT/10 -----	28
d.	Interface -----	28
2.	Software -----	28
a.	Language -----	28
b.	Segmentation -----	30
C.	COMPOSITION OF SYSTEM AUDIO OUTPUT -----	30
1.	Element Types -----	30
a.	Duration -----	30
b.	Amplitude -----	32
c.	Functions -----	32

d.	Frequency -----	32
e.	A.M. and F.M. Function Time Scales ----	32
2.	Main Sequence and Signal Definition -----	32
3.	Subsequence Group -----	34
D.	FREQUENCY ANALYSIS -----	35
III.	SYSTEM OPERATION -----	39
A.	SYSTEM COMMAND LANGUAGE -----	39
1.	Element, Define -----	40
a.	Functions -----	40
b.	Frequencies -----	40
c.	Amplitudes -----	40
2.	Find Spectrum -----	41
a.	Single Function -----	42
(1)	Options -----	42
(2)	Plot -----	42
b.	Product of Two Functions -----	42
3.	Define Signal -----	42
4.	Delete Signal -----	44
5.	Insert Signal -----	44
6.	Display Signal -----	45
7.	Print -----	45
a.	Amp, Print; Freq, Print -----	45
b.	Sig Seqn, Print -----	45
c.	Functions -----	45
8.	Store -----	48
9.	Recall -----	48
10.	Tape -----	48

11.	Drum -----	48
12.	Execute -----	48
	a. Rite -----	49
	b. Last -----	49
	c. Alternate -----	49
	d. Envelope -----	49
	e. Music -----	50
13.	Comment -----	50
14.	Clear -----	51
15.	Step -----	51
16.	Octave -----	51
B.	DATA STRUCTURE -----	51
	1. Element Type Data -----	51
	2. Main Signal Sequence Data -----	52
IV.	ANALOG COMPUTER CIRCUITS -----	56
A.	FUNCTION SEGMENT SLOPE SWITCHING -----	56
B.	SEGMENT BREAKPOINT STORAGE AND SWITCHING -----	56
C.	PERIODIC WAVE GENERATOR CONTROL LOGIC -----	59
D.	PERIODIC WAVE FUNCTION GENERATOR -----	61
E.	TIMING SIGNAL GENERATOR LOGIC -----	64
F.	PERIODIC WAVE FUNCTION BREAKPOINT DETECTOR ---	64
G.	SHIFT PULSE GENERATOR -----	67
H.	SEGMENT SHIFT REGISTER -----	69
I.	AM FUNCTION GENERATOR -----	69
J.	AM FUNCTION BREAKPOINT DETECTOR -----	73
K.	AM FUNCTION GENERATOR LOGIC -----	75
L.	FM FUNCTION GENERATOR -----	77

M.	FM FUNCTION GENERATOR LOGIC -----	79
N.	SIGNAL TIMER -----	81
O.	COMPONENT SETTINGS -----	83
V.	SYSTEM TESTING -----	85
A.	SPECIFIC EXPERIMENTS -----	85
1.	Music Synthesis -----	85
2.	Single Equivalent Formants -----	89
B.	GENERAL OBSERVATIONS ON TESTING -----	95
VI.	CONCLUSIONS AND RECOMMENDATIONS -----	98
	APPENDIX A: SYSTEM START UP -----	101
	COMPUTER PROGRAM -----	103
	LIST OF REFERENCES -----	195
	INITIAL DISTRIBUTION LIST -----	198
	FORM DD 1473 -----	199

I. INTRODUCTION

A. BACKGROUND

Sound communicates information. It is one of the most important and most widely utilized means by which humans communicate. But human communication implies not only the production and transmission of sound, but also the perception of sound. The accelerated progress of technology in the fields of electronics and computer science in the last decade has tended to produce a false confidence in our practical ability to duplicate or simulate human processes such as the perception of sound. The results have been disappointing and expensive. The perception of sound is one of the most interesting and mystifying of human processes and its complexity has been somewhat obscured by its intimate proximity with human existence. Additional knowledge about this fundamental human process will undoubtedly prove extremely beneficial to man.

As noted by Beardslee and Wertheimer [3], even though perception is one of the oldest areas of research, the work in perception has not yet yielded a very quantitative set of laws, and even though there is an emerging number of principles concerning the perceptual process, it is in many ways very incomplete. More directly related to the perception of sound, Green pointed out that the field of psychoacoustics is characterized by the lack of any integrative

structure from which to view the rapidly expanding literature [18].

"The human ear has been the object of numerous theoretical and experimental studies for centuries, and almost invariable each conceptual or experimental advance has put new obstacles in the way of understanding what was once thought to be a simple system. A survey of psychoacoustical literature of the past hundred years shows clearly how extremely sophisticated the auditory system is, and how new experimental facts appear to render the development of a complete theory of hearing more remote." [19]

One of the modern pioneers in the field of psychoacoustics, Harvey Fletcher, while recognizing the many aspects of the perception of speech sounds, divided the principle aspects into five categories - the interpretive aspect (recognition of sounds), the loudness aspect, the pitch aspect, the quality aspect, and the tempo aspect [14]. These groups can also be applied to sounds other than speech sounds.

The information conveyed by sound, however, is ordinarily not transmitted by isolated sounds, but by a complex sequence of sounds. The pattern of changes within a sequence in one or more of the aspects conveys additional information. In speech sounds, the juxtaposition of individual sounds, or phonemes, creates words, which are grouped to form phrases and sentences. Similarly in musical sounds, the connection of several sounds of changing pitch and tempo creates a melody. While one aspect of the sounds may remain fairly constant or simply be repeated, such as the pattern of pitch changes, other aspects such as tone,

tempo, or loudness may undergo changes so as to provide new information, rather than just repeating previous patterns (e.g. Ravel's Bolero).

A review of the literature dealing with the experimental analysis and synthesis of sound suggests three general types of sounds and sequences of sounds - language or speech, musical, and others not fitting the first two types. Within each of these three types, each of the aspects suggested by Fletcher has been, and is, fertile area for investigation and experimentation. In speech sounds, the emphasis has been on the quality and interpretive aspects, since the information is largely a function of the formant structure of the sounds, with secondary information being supplied by the loudness, pitch, and tempo aspects. In musical sounds, the emphasis has been on the aspects of quality, or timbre, and on pitch.

The classification of sounds by types and quantitative descriptions of the aspects of sounds are somewhat arbitrary and subjective. Speech distorted with noise might be classified as either speech or noise depending both on the objective degree of distortion and on the subjective ability of a listener to extract the speech information. The drums used in African drum language are capable of transmitting two distinct tones or pitches, which are not used to transmit vowels or consonants but rather to mimic the sequence of tones of well known stock phrases [7]. One who is unaware of the underlying language being mimicked

would classify the sounds as either noise or music rather than language. Moorer also addressed the general problem of distinguishing between sequences of speech and musical sounds [22]. He concluded that the representation of music by a type of grammar is similar to representing speech by grammar and suggested that this is an indication that the production mechanisms are actually closely related. He also made distinctions between speech perception and the perception of music, pointing out that in speech the pitch is of little consequence, whereas in music, pitch and rhythm are the essence of the piece. While such distinctions may be valid in comparing American music with the English language, it does not appear that they are universally valid. Certainly, there are some languages, such as Chinese, where pitch plays an essential role. One can also find examples of music where pitch plays a minor role and quality of tone and rhythm are of primary importance (e.g. One Note Samba).

While a cursory examination of such comparisons between speech and music might give one the impression that such pursuits have little practical value, additional investigation and reflection yield more meaningful and productive insights and questions. For example, consider the following.

"The melody sensation seems to be invariant over a large class of note-to-sound transformations. Melodies can be played on instruments that have quite different frequency transfer characteristics, and still be recognized. Some instruments produce sounds with a rich spectrum comprising the fundamental and many partials (stringed instruments); some have little else than

the bare fundamental (flute in high register); others generate spectra having a strong "formant" region with the result that the lower notes usually lack the lower partials, including the fundamental (oboe)." [19]

That is, one can translate the pitch of a musical melody and present the melody with a variety of spectrally differing instruments, or with a transformation of the spectral formant structure, and somehow the human perception mechanism is able to extract the information about the melody. In the perception of speech, a translation of pitch does not prevent a listener from detecting the information, and yet a translation of the entire formant structure produces a different effect. Doubling or tripling the frequency of all spectral components, such as by speeding up the playback of a record or tape recording, produces less recognizable audio presentations. A similar effect occurs with naval divers using a helium-oxygen atmosphere, where due to the increased pressure and the change in the chemical composition of the atmosphere, the frequency transfer characteristics of the vocal mechanism are altered to produce speech with transformations of the usual formant structure. This produces speech that is not readily understood. If the same, or parts of the same, perceptual process are involved in both instances, it would be very beneficial to determine why transformations are possible in one case and not the other, and how the audio presentations could be altered or the listener conditioned to make both presentations equally understandable.

As another example, consider the perception of the sound of several speakers, speaking different words simultaneously. When the several voices are combined and presented as a monophonic source, recognition of information becomes difficult or impossible. In music, however, polyphonics in either monaural or stereo forms do not prevent the recognition of information such as melody or instrument tones. An observer of polyphonic music is able to listen to several melodies simultaneously and is also able to consciously concentrate on only one of the melodies to a degree not possible in simultaneous speech.

Birds produce sounds that appear to be more musical than speech-like, and yet some birds have the ability to mimic human speech. The transfer characteristics of the vocal tracts of such birds are obviously not the same as those of the human vocal tract. Comparisons of the formant structures of the human and bird speech sounds might reveal different formant structures that are perceptually equivalent and provide insight as to why. Such investigations might be related to Focht's work in the machine recognition of speech, which is based on a technique of transforming multi-formant speech sounds to single equivalent formants [15].

The communication system used in the perception of sound is obviously extremely complex.

"The interaction of the physical-acoustic phenomenawith the human intellect is a complicated one. Perceptions of rhythm, nuance, sound quality, pitch, etc. depend on several physical parameters in a complex way and are interrelated with the physiology and psychology of the listener." [28]

It is, perhaps, the complexity and the vast number of unknowns in the field of sound perception that make its investigation both mystifying and interesting. And it is the importance of sound in human communications and the similarities and differences in speech and musical sounds that make experimentation and investigation in both areas worthwhile.

The rapid advancement of computer technology during the past decade has provided increased opportunity for application in the field of sound, so that computers now function as some of the most powerful tools for the experimental analysis and synthesis of sound. Analysis and synthesis of sound are important steps in the investigation of man's perception of sound, and are prerequisites to understanding that perception. In the many areas of sound investigation, analysis has attempted to determine a physical description of the sound; and synthesis has been employed to test the accuracy or adequacy of the physical description. Synthesis can be used to test the significance or insignificance of various parameters of almost any physical description of a sound. With regard to musical instrument sounds, Mathews and Risset have pointed out that in most cases in the past, the synthesized sounds bore little resemblance to the actual

tones [24]. This was due mainly to the fact that most earlier attempts at analysis were directed mainly at a steady state frequency analysis, which is understandable in view of the less developed technology of the past. Today, however, the vastly improved analog and digital tools available have made not only better steady state but also transient analysis and synthesis practical, and have opened the door for the development of new and better techniques and applications. For example, Freedman has presented a technique for the analysis of musical instrument tones that is also applicable to other acoustical phenomena such as electroencephalogram, electrocardiogram, and geodynamic signals [16]. Slaymaker has conducted some interesting synthesis experiments with musical chords using tones having stretched partials, which have questioned the adequacy of traditional theories of consonance and dissonance.

"All of the traditional predictability was gone. The chords sounded smooth and nondissonant but strange and somewhat eerie. The effect was so different from the tempered scale that there was no tendency to judge intuneness or out-of-tuneness. It seemed like a peek into a new and unfamiliar musical world, in which none of the old rules applied, and the new ones, if any, were yet undiscovered." [26]

The growing ease of creating and experimenting with new scales and tones using computer synthesis may prove to accelerate man's musical evolution and to broaden and illuminate human perceptual abilities as yet unused.

Analog musical synthesizers such as the MOOG, the TONUS, and the BUCHLA BOX are finding increasing application in

modern music. Hybrid applications such as Ashton's interactive organ-computer communications network [2] are also developing.

Modern techniques for the analysis and recognition of speech [8,20,21,27] would not have been possible without the digital and analog tools available today. The application of these tools has also shown how the problem of machine recognition of speech is much more complex than originally imagined. Techniques for speech synthesis have progressed in a similar manner with the technological advances [13,25].

While the digital analysis of speech and music has certainly come a long way, it does not approach the ease with which the human perceptual process analyzes sounds. And while digital and analog techniques for the synthesis of sound can create remarkably accurate imitations of speech and familiar musical sounds, the ease of use and the ability to practically apply these techniques leave a great deal of room for development. Additional experimentation with both speech and musical sounds is required. Hopefully this experimentation may provide additional data from which more meaningful and useful quantitative theories of perception may be derived; and these theories may, in turn, return information leading to newer and improved techniques and applications. It was in this spirit that this work was begun.

B. PURPOSE AND APPROACH

The goal of this work was to develop an interactive hybrid computer system for the real-time definition and synthesis of simple approximations of periodic and quasi-periodic audio frequency waveforms. Such a system would be a worthwhile electroacoustical tool for use in the study of psychoacoustical phenomena. A system with an interactive mode of operation would allow the user to vary parameters of the audio waveforms, or sequences of waveforms, and then immediately make perceptual observations and evaluations. By giving the system a frequency analysis capability, comparisons between changes in the time domain and resulting changes both in the frequency domain and the domain of perception are possible. These features would make the system not only a worthwhile tool for experimentation but also a valuable tool for instruction.

Another objective of this work was to test some hybrid techniques, using the facilities of the Naval Postgraduate School Electrical Engineering Department's Computer Laboratory, for generating the sequences of waveforms. This work approaches the job of synthesis from a time domain perspective. Where the waveforms to be synthesized are simple and periodic, such an approach can be more useful than one that is frequency domain oriented. Obviously, a triangular waveform might be better synthesized with several first order segments rather than by trying to provide the individual spectral components. There are also some more complex

sounds that can be described more easily and meaningfully in the time domain. Some musical instrument sounds (e.g. organ sounds) can be characterized by a simple periodic waveform, or carrier, that is amplitude modulated by another simple function. The envelope of the carrier is apparent from a time domain inspection of the signal but may not be so apparent from an inspection in the frequency domain. Periodic audio signals that are characterized in the frequency domain by lower harmonics of small magnitude with a formant structure in the higher harmonics are most often viewed as resulting from a filtering process, or a multiplication in the frequency domain. Some signals of this type can, however, be synthesized by translating a formant structure centered around zero frequency to a different center frequency by a process of amplitude modulation, or multiplication in the time domain.

In general, most synthesis techniques have been oriented toward a frequency domain perspective; that is, based on a previous spectral analysis, an attempt is made to generate a waveform containing the "important" spectral components. The resulting digital description of the synthesized signal has usually been a sequence of equally spaced zero order segments. Another technique, and that pursued in this work, is to directly generate an approximation of a time domain waveform using a function made up of first order segments of varying length.

C. PROBLEMS AND SOLUTIONS

The technique of time domain synthesis using unequally spaced first order segments presented several problems. Non-monotonic function generators that were both easily and rapidly programmable by the digital computer were not available. Therefore, one problem was to find a means to generate the functions. As a solution, function generators for this work were developed using the general purpose analog computer components available in the Computer Laboratory of the Naval Postgraduate School Electrical Engineering Department.

Three types of function generators are used in the system; the primary differences among the three involves the methods of supplying inputs to the generators. In the first type of function generator, one D/A channel is used for the segment slope and another for the breakpoint. A slope and breakpoint are transmitted to the analog computer and stored by analog storage devices. The slope and breakpoint values of the next segment are then transmitted over the same channels (the channels themselves then act as the analog storage devices for the next segment). Thus, when switching from one segment to the next, no action is required by the digital computer, which has the time between segments to transmit the next segment values.

The second type of function generator uses two D/A channels for the current segment slope and breakpoint values and two different channels for the values of the next segment.

Again, when switching from one segment to the next, no action is required by the digital computer. The advantage of this method is that switching is faster because no time is required for the storage devices to track the new values; however, two additional D/A channels are required.

In the third type of function generator, the slope of each segment is transmitted on a separate channel and then stored by an analog component. The segment breakpoints are then transmitted on the same group of channels. The advantage of this method is that once the function description is transmitted to the analog computer, no action is required of the digital computer between segments.

Another problem was the limited number of analog computer components and D/A channels available. Part of the solution was the decision to use the three different types of function generators, each of which is used to generate a different type of function. The D/A channels and analog components for each type were allocated on the basis of expected function period. Less channels and components were allocated for the generators of functions with longer periods since the longer periods allow analog computer data to be transmitted on an "as needed" basis. But, because there were still not enough D/A channels for the parallel transmission of all the analog information needed, the transmission process was broken down into a series-parallel operation, using analog devices to store earlier parts of the transmission. Since the number of D/A channels and analog computer

components required depends on both the number of function generators and the number of segments in a function, a decision was made to limit the number of segments in a function to eight. This decision also involved a compromise between the storage capability of the digital computer, the number of function descriptions (and their sizes) stored by the digital computer, and the amount of digital computer storage required for the system's programs.

Even though the periods of the functions generated are in the audio range (and usually limited to a maximum period of one millisecond), there was still a problem in switching from one segment to the next. Although the average segment time (at a one millisecond function period and eight segments per function) is 125 microseconds, individual segment times can be much less --- at the extreme, a segment with infinite slope has a zero time. The general purpose analog computer components used for storage and switching from one segment to the next have a finite recovery time which must not exceed the time between segment breakpoints. Therefore, restrictions (enforced by the digital programs) were placed on the magnitudes of segment slopes.

In the early stages of testing the system, it was found that transients in the analog circuitry during the recovery time often resulted in false detection of breakpoints. To solve this problem, additional circuitry was devised to disable the breakpoint detectors until component recovery was complete.

Another difficulty associated with using general purpose analog computer components is the limited usable voltage range of these components. The analog computer available in the Computer Lab has a linear range of -100 to +100 volts. This, together with the 15 bit accuracy of the D/A converter placed restrictions on the range of values of segment slopes and breakpoints. These restrictions are recognized by the system's digital programs. When a user defines a function, the system checks the definition to insure that the restrictions are not violated.

Considerations of technical and economic feasibility have in the past necessitated delegating certain aspects of audio analysis and synthesis to specialized hardware which is controlled by a digital computer. For example, in speech synthesis, an electronic model of the vocal tract is usually implemented with special hardware. The model's transfer characteristics are then controlled by a digital computer to generate the complex analog signals that represent speech. The function generators for this work were developed using the general purpose analog computer components available in the Computer Laboratory. However, the status of solid state technology is such that specialized hardware for direct time domain synthesis could be developed.

Another significant obstacle to using the technique of this work is the lack of psychoacoustical data on how much approximation is acceptable in the time domain (in producing

perceptually accurate or adequate sound). This represents an unsolved problem; however, the system itself will be useful in investigating this area.

II. SYSTEM DESCRIPTION

A. GENERAL SPECIFICATIONS

1. User Oriented

It was intended that the system developed be as user oriented as practical, within the constraints of the available hardware, with the objective of enhancing the intelligent involvement of the user with the system's operation. This required that communications between the user and the system be reasonable rapid and convenient. Therefore, a graphics terminal was chosen as the primary means of input and output to facilitate operation in a real-time interactive mode. The graphics terminal allows inputs to be either typed or entered on the terminal CRT with a light-pen and it displays information on the CRT in textual and pictorial forms. The graphics approach is advantageous not only from the point of view of speed and convenience, but also because of its ability to display pictorial information, which is often perceptually more informative than numerical data.

A user oriented input language was incorporated to make the system's use more convenient and understandable. A limited vocabulary of easily understood command words allows the user to conveniently instruct the system to perform specific operations. The system converses with the user by displaying the possible operations or commands from

which the user may choose, and by prompting the user to respond with an input.

Efficient utilization of the system required that it be capable of recovery from input errors. Invalid command words or numerical inputs outside of allowable ranges are, therefore, ignored by the system, which does not proceed until a valid input is received.

2. Permanent Records

While the graphics terminal provides speed and convenience, it lacks the ability to provide permanent records for future use or study. Therefore, the system provides a means for generating permanent and semipermanent records using the line printer, magnetic tape units, and magnetic drum. The line printer is used to create plots of pictorial data (originally displayed on the CRT) and to print tabulations of numerical data. Magnetic tape and drum are employed to store a user's definitions for later use.

3. Simple Time Domain Models

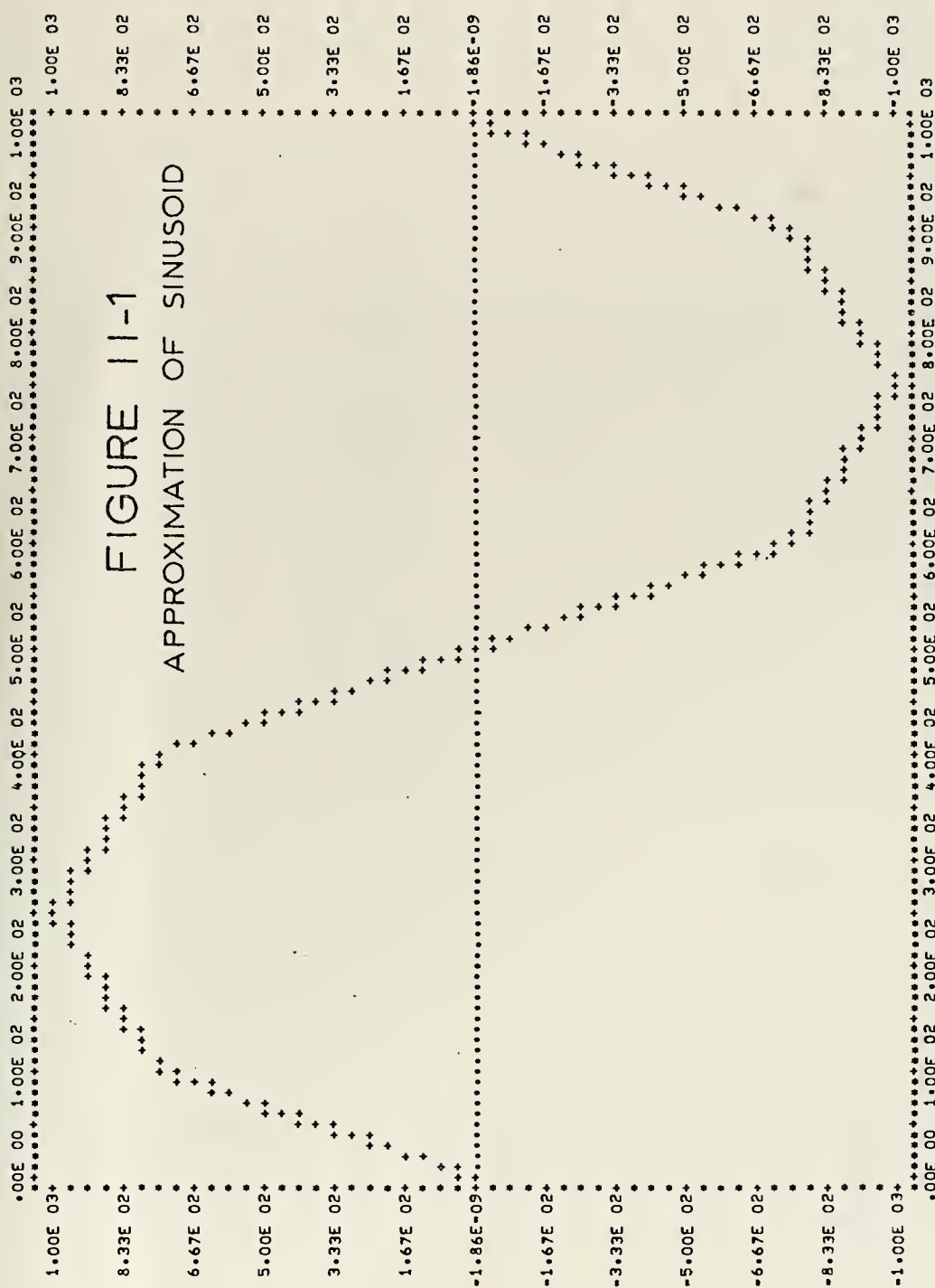
The system uses simple time domain models of periodic functions. There were several reasons for this decision. First, some signals, or characteristics of signals, are more easily or better described in the time domain than in the frequency domain. Specifying the coordinates of the breakpoints of a square or a triangular waveform is easier and more descriptive (perceptually) than providing the Fourier series for the waveform. Characteristics such as attack and decay of musical instrument sounds or the formant

trajectories of speech sounds are more meaningfully described in the time domain. The system uses simple models that consist of eight connected straight line segments to approximate more complex time domain waveforms. Figure II-1 illustrates such an approximation of a sinusoidal waveform. Simple models are, naturally, easier to describe. Since less data is needed to describe them, data storage is more economical, data transmission (during the process of synthesis) is reduced, and data processing time is decreased. This facilitates real-time interactive operation. Several signals using different simple models can be connected together to create a sequence of signals and models can be multiplied together to modulate the amplitude envelope of one of them. A final reason for the choice of simple models was the limitations imposed by the amount of analog and digital hardware available.

While it is possible to determine the physical accuracy of approximating a more complex waveform with a simple model, such an analysis does not describe the perceptual accuracy or adequacy of the model; that is, how well or how adequately the simple model imitates the perceived sound of its more complex counterpart. It was intended that the system could be used to investigate this.

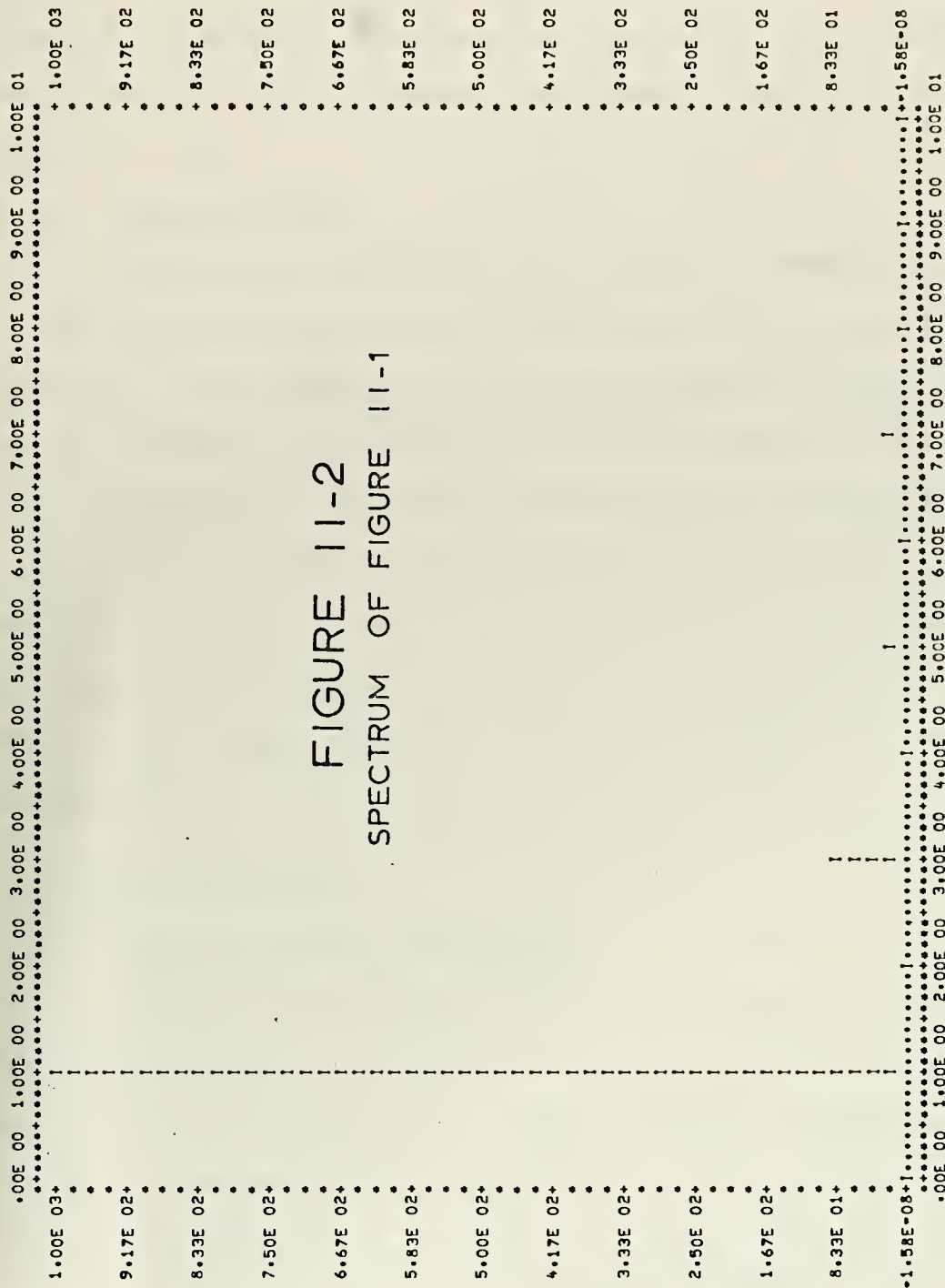
4. Frequency Analysis

The system provides a limited degree of frequency analysis of the signals created with the models defined by the user. This allows the frequency spectrum to be related



X(1)=	0	X(2)=	100	X(3)=	250	X(4)=	400	X(5)=	500	X(6)=	600	X(7)=	750	X(8)=	900	X(9)=	1000
Y(1)=	0	Y(2)=	750	Y(3)=	1000	Y(4)=	750	Y(5)=	0	Y(6)=	-750	Y(7)=	-1000	Y(8)=	-750	Y(9)=	0

PERIODIC WAVE NUMBER 1
 FREQUENCY SPECTRUM MAGNITUDE • NUMBER OF POINTS • 11



to the audio presentation of the synthesized signal, and it provides a tool for comparing changes in the time domain parameters with the corresponding changes that occur in the frequency domain.

5. Signal Output

The actual synthesis of the signals is accomplished by the analog computer, under the control of the digital computer. The analog voltage representing the signal output is available for viewing on an oscilloscope or for connection to an audio amplifier (for immediate observation) or to a tape recorder (for a permanent audio record). Minor changes in analog computer components allow time scaling of the signals to facilitate analog plotting of the signal output and allow major changes in the dynamic range of signal parameters.

B. SYSTEM COMPONENTS

1. Major Hardware Components

The major components of the Electrical Engineering Department Computer Laboratory that are used in the system are briefly described below. More detailed information may be found in Refs. 1, 9, 31, and 32.

a. XDS 9300

The XDS 9300 is a high-speed, general purpose, digital computer with a 1.75 microsecond cycle time. It uses 24-bit words and has a 32K word core memory. It is supported by a magnetic drum with a 524K word capacity, two

magnetic tape units, a card reader, a high-speed line printer, and a teletypewriter.

b. COMCOR CI-5000

The COMCOR CI-5000 is an analog computer with a large selection of analog elements and a full complement of digital logic.

c. ADAGE AGT/10

The ADAGE AGT/10 is a small general purpose digital computer using 15-bit words with an 8K memory. Typical instruction times run five to six microseconds. The AGT/10 incorporates a cathode ray tube (CRT) for the display of graphics and text, and is further supported by a teletypewriter and a magnetic disc for secondary storage.

d. Interface

The three major components above are interfaced by a high speed data channel and conversion equipment. The transfer rate is 250,000 words/second to the XDS 9300, and 160,000 words/second from the XDS 9300.

2. Software

a. Language

The programs for the system were written primarily in XDS FORTRAN IV, although some of the programs contain a mixture of FORTRAN and assembly language, which is permissible in XDS FORTRAN IV. Assembly language instructions were incorporated primarily where speed of operation was considered critical to the operation of the system.

b. Segmentation

Since the length of the system's digital program exceeded available core, the program was segmented and overlaid as shown in Figure II-3 below.

c. COMPOSITION OF SYSTEM AUDIO OUTPUT

In this section, the audio output of the system is described. The output consists of a sequence of user defined signals. Basically, each signal consists of a periodic wave function, analogous to a carrier, that is amplitude modulated and frequency modulated by other functions. The duration, amplitude, and the frequency of the periodic wave function, as well as the function itself, are defined by the user. The amplitude and frequency modulation functions, and their periods, are also defined by the user. Individual signals are then joined to form a single main sequence. Up to 100 subsequences of the main sequence may be joined in any order and synthesized.

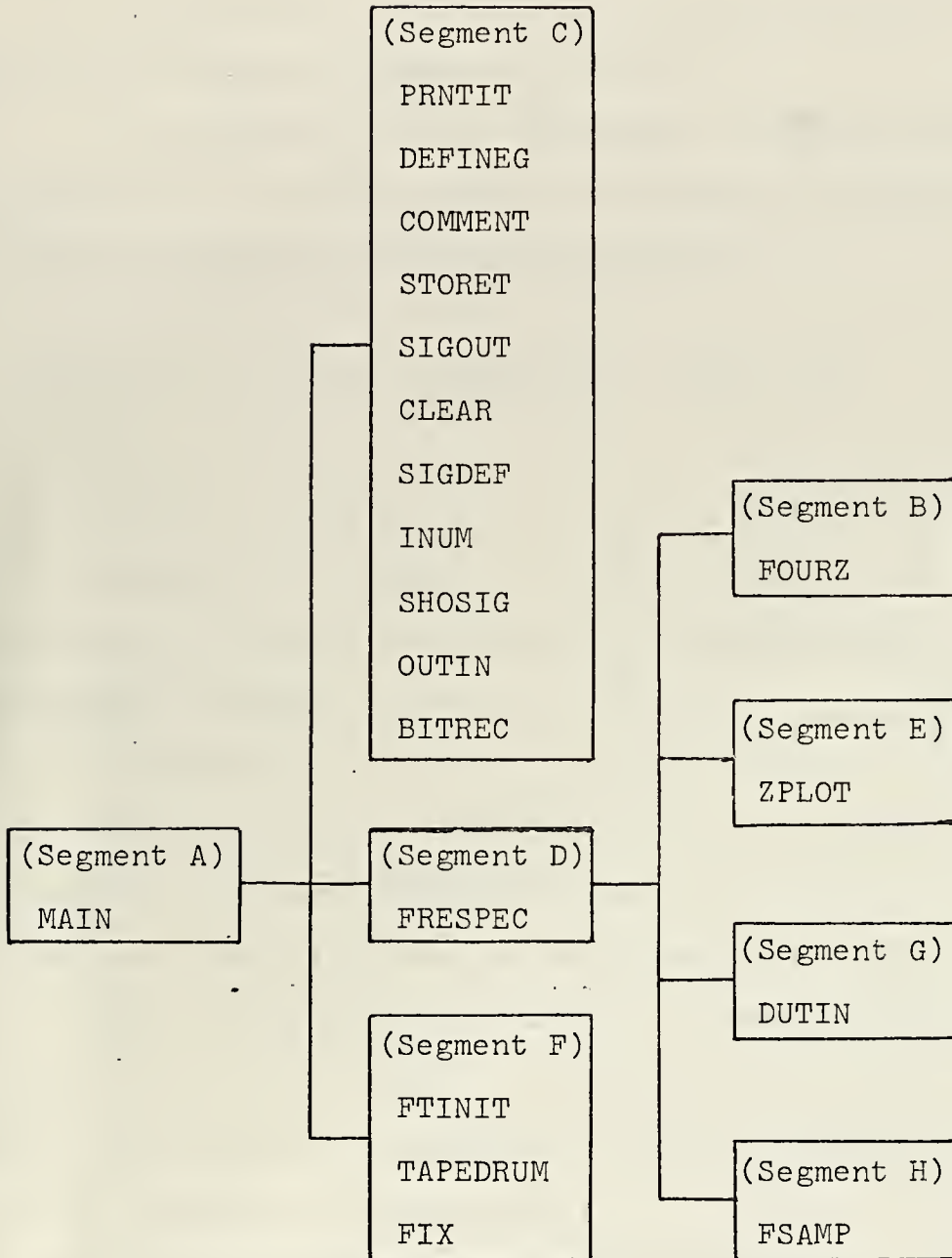
1. Element Types

There are eight types of elements. The elements of any one type may differ among themselves, but still have a common general form. Individual signals are defined by one element of each of the types. The element types are described below and summarized in Table II-1.

a. Duration

The elements of this type are used to specify the time length of a signal.

FIGURE II-3 PROGRAM SEGMENTATION



b. Amplitude

The elements of this type are used to specify the relative audio strength of a signal.

c. Periodic Wave Function, Amplitude Modulation Function, and Frequency Modulation Function

The elements, or the functions, of each of these types are made up of eight connected straight line segments. The functions conform to the following form.

$$f(t) = c + \sum_{i=1}^8 a_i \cdot [t-d_i][U(t-d_i) - U(t-d_{i+1})]$$

where T = period, $d_1 = 0$, $d_9 = T$, $d_i < d_{i+1}$, and U = the unit step function. An example is shown in Figure II-1. The functions are defined and stored in a non-frequency, or time normalized format.

d. Frequency

The elements of this type are used to specify the frequency of the periodic wave functions.

e. A.M. Function and F.M. Function Time Scales

The values of the elements of these types are specified at the time a signal is defined and they are used to specify the relative frequencies of the A.M. function and the F.M. function of that signal.

2. Main Signal Sequence and Signal Definition

The main signal sequence consists of one or more user defined signals, each described by the eight element types previously discussed. As each individual signal is defined

TABLE II-1 SUMMARY OF ELEMENT TYPES

Element Type	Number Available	Parameter Range	Parameter Units	Notes
Periodic Wave Function	10 (0-9)			8-segment function; I.C. = 0.
A.M. Function	10 (0-9)			8-segment function.
F.M. Function	10 (0-9)			8-segment function; I.C. = 0.
Frequency	72 (1-72)	100-9999	Hz X 10	Frequency of Period Wave Function.
A.M. Function Time Scale	*	100-9999	Hz X B	B is variable; usually set at 0.02 with analog components.
F.M. Function Time Scale	*	100-9999	Hz X C	C is variable; usually set at 0.005 with analog components.
Amplitude	10 (0-9)	10-9999		Relative signal amplitude.
Duration	*	10-4000	ms	Signal duration. Can be scaled with analog comp.

* Not predefined. Value specified when signal is defined.

by the user, it is placed after the previous signal to form a single main sequence. From this main sequence, a part (or subsequence) or parts (a group of subsequences) may be selected for synthesis.

There are two ways in which the actual definition of a signal may be made by the user. The first way is by specifying one of each of the types of elements. The second way is by specifying less than eight of the element types. When the second way is used, the system assumes that the elements of the types not specified are to be continuations or duplicates of the corresponding types of the previous signal synthesized.

There were two reasons for incorporating this distinction into the system. First, it reduces the amount of data that must be entered and stored to describe a signal when some of its elements are the same as those of the previous signal. Second, when the element type continued is one of the functions, it allows that element to continue uninterrupted from one signal to the next.

3. Subsequence Group

When the user elects to execute the actual synthesis, he may specify up to 100 subsequences of the main sequence. The subsequences will then be synthesized in the order specified. Thus a long string of signals that has repeating parts may be created at execution time without the need for repeating the definitions of the duplicated parts in the main sequence. The grouping information is not stored permanently,

however; therefore, each time the synthesis is executed, the grouping information must be respecified.

D. FREQUENCY ANALYSIS

For computing the frequency spectrum of the periodic wave and the A.M. functions, an FFT (fast Fourier transform) version of the discrete Fourier transform is utilized. Since the functions used are considered to be periodic, their Fourier series expansions are of the form

$$x(t) = \sum_{n=-\infty}^{\infty} c(n) \exp[2\pi i(nt/T)]$$

where the $c(n)$ are given by

$$c(n) = 1/T \int_0^T x(t) \exp[-2\pi i(nt/T)] dt$$

and T is the period of the function.

The function $x(t)$ may be sampled at N equally spaced points between 0 and T to generate the sequence $x(j\Delta t)$, where $\Delta t = T/N$. The equation for the sampled function is then

$$x(j\Delta t) = \sum_{n=0}^{N-1} c_p(n) \exp[2\pi i(nj/N)] .$$

Using the properties of the discrete Fourier transform [10], it can be shown that the function $x(j\Delta t)$ has the finite Fourier transform $c_p(n)$, where $c_p(n)$ is given by

$$c_p(n) = \sum_{k=-\infty}^{\infty} c(n + Nk) .$$

Since the exact Fourier series is infinite, it is necessary to make N large enough so that the aliasing error introduced by approximating $c(n)$ by $c_p(n)$ is reduced to an acceptable level. When this is done, then $c_p(n) \approx c(n)$ for $n = 0, 1, 2, 3, \dots, N/2$.

As discussed in Ref. 6, the errors that are possible in calculating the discrete transform are those due to aliasing, those due to the finite precision of the computer, and "leakage" errors. An accumulated computational error occurs in evaluating the transform due to the imprecision with which the data and the intermediate computational results are represented in the computer. Reference 17 explains that this error is significantly reduced by the utilization of the FFT and floating point arithmetic; the FFT algorithm computes the discrete Fourier transform using only $2N \log_2 N$ operations, rather than the N^2 operations previously required for the direct evaluation of the discrete Fourier transform.

As discussed in Ref. 6, if the sample period of the function being transformed is truncated in such a way that it contains cisoidal components that do not repeat an integral number of times in the sample period, or are discontinuous at the end of the sample period, a "leakage" error will occur in the frequency analysis. This problem can occur when using the system to find the spectrum of a

periodic wave function amplitude modulated by an A.M. function when the ratio between the periods of the two functions is not an integer value.

To generate samples for the FFT, the system uses the coordinates of the breakpoints of the segments of the function to calculate the slope of each segment. It then calculates function values between the breakpoints. When a single function is to be transformed, 512 points are used in the sample period. When the transform of a periodic wave function multiplied by an A.M. function is to be calculated, the periodic and A.M. functions are sampled individually, the samples are multiplied, and the transform is then taken. This eliminates the need for a complex convolution in the frequency domain. For this case, a sample size of 1024 is used.

An important determination in the application of the discrete Fourier transform is that of the sample period. This does not present a serious problem when the signals being considered are steady state or when the spectral statistical properties of the signal are reasonably constant over a determined sample period (and the average spectrum is what is sought). However, where the various spectral components change with time, this can be a serious problem. When the signals represent sound, the problem is even greater because the Fourier analysis must have meaning with respect to the perceived sound. In sound, one is often interested in how the frequency spectrum changes as a function of time

(with a particular sample period). Until more is known about the process of human perception of sound, this will continue to be an area for investigation.

While frequency analysis was not intended to be the major purpose of this work, the problem described above was recognized. The system's programs do allow the user to set the period over which a signal is analyzed so that changes in period can be related to changes in spectrum and changes in perceived sound.

III. SYSTEM OPERATION

A. SYSTEM COMMAND LANGUAGE

The user converses with the system, via the graphics terminal, by using special system command words. The system displays a list of alternatives from which the user may choose and prompts the user to indicate his choice by entering a specific command word followed by a C/R (carriage return). The system then responds by either performing the indicated operation or by providing additional directions to the user and requesting additional inputs. Command words that are not a part of the system's vocabulary are ignored. All data is entered by the user in integer format and terminated by a comma and a carriage return; decimal values must be multiplied by a scale factor (given by the system during use) to make them integer valued.

The operation of the system can be best explained by a discussion of the consequences of entering specific command words. Initially, the user is presented with a display of primary operations and is asked to make a choice. The results of choosing each of the primary operations is discussed below. In each case in the following descriptions, command words and other parameters are entered by the user at the explicit direction and guidance of the system. To terminate an operation, the user enters the special command word END.

1. Element, Define

The system responds by displaying a list of five types of elements the user can predefine (periodic, A.M., F.M. functions, amplitudes, and frequencies) and requests the user to choose one of the types.

a. Functions

If the type chosen is one of the functions, the system asks the user to specify the number of the element (0 through 9). When a valid number is received, the system displays the function selected and asks the user whether he wishes to modify the function by using the light-pen or by typing the coordinates of the breakpoints of the segments of the function (initially the functions are initialized to the pseudo-sinusoidal waveform shown in Figure II-1). The user responds by typing a T for typed input, an L for light pen input, or a C/R to erase the display and select a different numbered element. When typewritten input is selected, the system displays the X and Y coordinates of the function segment breakpoints along with the graphical display of the function and prompts the user to input the coordinates of each breakpoint. As each X and Y value is typed, the coordinate list and the function-display are updated (if the user does not desire to change an X or Y value, he may type a C/R and the system advances to the next value). If, in the process of modification, the user wishes to copy the displayed values of all remaining X and Y values, he types the letter C.

When the user has completed the definition, the system examines the definition for errors (such as segment slope magnitudes outside of allowable ranges). If errors are detected, the system restores the original function and the user must reenter his definition. If no errors are detected, the system normalizes the X values of the function so that the largest is 1000 units. The user also has the option to normalize the Y values.

At this point, the user may change his definition by again typing T or L as discussed before.

b. Frequencies

When the frequency type of element is chosen, the user is presented with a display of the current values of the 72 frequency elements and is prompted to select one of the elements and then enter its new value. As each of the frequency elements selected is defined, the display is updated and the user is requested to select another element.

c. Amplitudes

The process of defining, or changing, amplitude elements is similar to that of the frequency elements, except that the current values are not displayed.

2. Find Spectrum

The system responds by asking the user whether he wants to find the frequency spectrum of a single function or of a periodic wave function amplitude modulated by an AM function. Depending on his choice, the following occurs.

a. Single Function

Prompted by the system, the user then chooses either a periodic wave function or an AM function and specifies an element number (0 through 9). After valid inputs are received, the system calculates the magnitude and the phase of the discrete frequency spectrum. The system then asks the user how many values, or points, of the calculated spectrum he wishes to display (he may choose between 5 and 399 points). A graphical display of the spectrum is then presented on the CRT. If the function being analyzed is considered to be periodic, each spectrum value displayed corresponds to an harmonic of the function, with the first value corresponding to the dc component. Being able to specify the number of points of the spectrum to be displayed is useful when the relevant information in the spectrum is concentrated in the lower harmonics. If the total calculated number of points were always displayed, the significant information might be bunched so closely in the display that it would be too difficult to view.

(1) Options. The user has several options in finding the frequency spectrum of functions. By following system direction he may set the period over which the function is analyzed to a value less than a full period of the function. He may also set the period of analysis equal to the period of the function, but truncate the function at a point less than a full period. A combination of these options is also possible.

(2) Plot. After a display of the frequency spectrum is presented, the system asks the user if he wants a line printer plot of the spectrum magnitudes being displayed. A sample plot is shown in Figure II-2. If the user elects to simply continue, the system again asks the user how many points are to be displayed.

b. Product of Two Functions

The system asks the user which periodic wave function is to be modulated by which AM function, and the ratio between the two. Operation then proceeds as for single functions.

3. Define Signal

The system responds to this choice by presenting the description of the last signal defined on the left half of the CRT and a list of the eight element types to be specified for the new signal on the right half of the CRT. The user may specify each new element in one of four ways.

In the first way, the user specifies the element by its number (if it is one of the predefinable types) or by its value (if it is an AM or FM time scale or duration element).

In the second way, the user types a C/R. When this occurs, the element is assumed to be a continuation of the corresponding element type of the previous signal. This makes specification of the new signal relative to the previous signal, and changing the previous signal would also change the new signal. During synthesis, it is relative to the previous signal synthesized.

In the third way, the user types the letter R and the system copies the specification of the corresponding element type of the previous signal defined. It often is easier to type an R than to type element numbers or values. Once an element is specified in this way, changing the previous signal will have no effect on the new signal.

The fourth way is by typing the letter C. It is equivalent to typing a C/R for all element types not yet specified for the new signal.

When all of the elements of the new signal have been specified, the new signal definition replaces the old display on the left half of the CRT and the user is invited to define another signal.

4. Delete Signal

This choice is used to delete a signal from the main sequence of user defined signals. The user is asked to identify the number (within the main sequence) of the signal, and the system responds by deleting the signal and decrementing the numbers of all successive signals by one. The system then invites the user to select another signal.

5. Insert Signal

This choice is used to insert a new signal between two previously defined signals in the main sequence. Following system directions, the user identifies the old signal after which the new signal is to be inserted. The old signal is then displayed on the left half of the CRT and the

user defines the new signal to be inserted in the manner described under Section III-A-3 (Define Signal).

6. Display Signal

The user identifies a previously defined signal and the system responds by displaying the element specification of the signal on the CRT. The user is then invited to identify another signal.

7. Print

When this choice is entered, the system responds by displaying a list of secondary choices. The results of selecting each are discussed below.

a. Amp, Print; Freq, Print

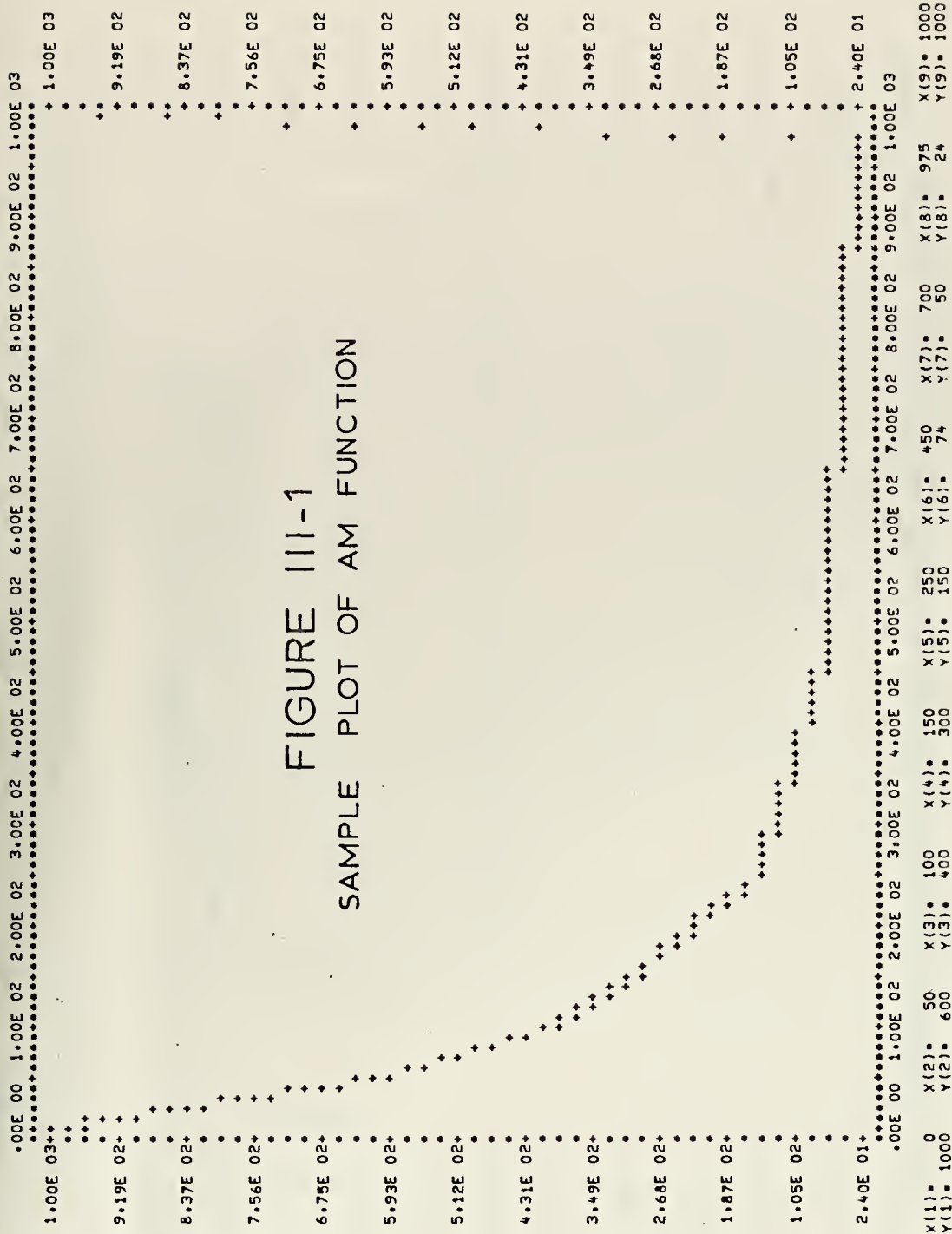
When one of these choices is selected, the system produces a listing of the values of the elements of the type chosen on the line printer.

b. Sig Seqn, Print

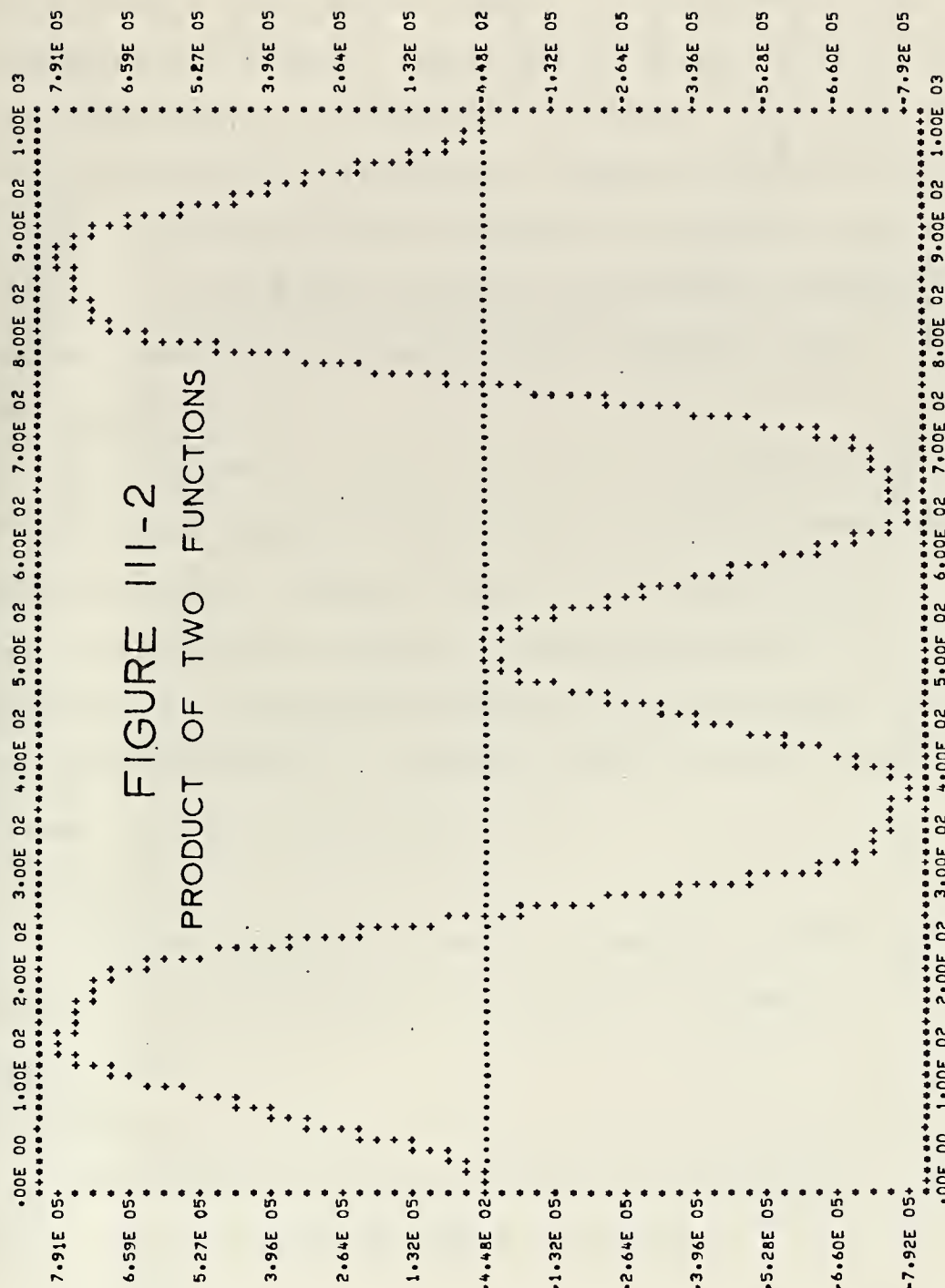
This choice is used to obtain a line printer listing of the descriptions of one or more of the signals in the main signal sequence. The listing is produced after the user inputs the first and last signals whose definitions are to be listed.

c. Functions

The last four secondary choices are used to obtain line printer plots of any of the eight-segment functions. The last choice is used to obtain plots of a periodic wave function modulated by an AM function. Sample plots are shown in Figures III-1 and III-2.



PERIODIC WAVE NUMBER 1 X A.M. FUNCTION NUMBER 1 RATIO = 2.00



8. Store

The Store command is used to transfer user definitions of signals and signal elements to the magnetic drum for semipermanent storage. Space for 25 files, or 25 sets of user definitions, is reserved on the drum. After the user specifies the file number, the system writes the information on the drum without removing it from core, and then returns to the list of primary operations. The file may be recalled later or transferred to magnetic tape.

9. Recall

This choice is used to recall all or part of the user definitions stored in one of the previously discussed files on the drum. The user may recall the main signal sequence or any of the element definitions stored in the file by following system directions, and could select different element types from different files and then store the combination in a new file.

10. Tape

The Tape command is used to transfer the files of user definitions on the drum to permanent storage on magnetic tape.

11. Drum

This command is used to read files stored on magnetic tape and transfer them to the drum.

12. Execute

The Execute command initiates the actual synthesis of the user defined signals. Any number of the following options may be typed next.

a. Rite

Just prior to synthesis, a listing of the main sequence numbers of the first and last signals of each subsequence is written on the line printer.

b. Last

Ordinarily when the last subsequence of a group has been synthesized, the entire group is repeated until the user halts the synthesis process. However, when the Last option is specified and when the last subsequence has been synthesized, only the last subsequence will be repeated.

c. Alternate

When a group of subsequences has been selected for synthesis and the Alternate option is specified, the last subsequence of the group is alternated with each of the other subsequences during execution. For example, the last subsequence may be one in which the functions of all of the signals are not directly specified (but are continuations of the functions of a signal in a different subsequence). The other subsequences could consist of only one signal each, with the functions directly specified. This would result in the synthesis of several sequences (each a combination of one subsequence and the last subsequence) that were the same except for the first signal and the functions.

d. Envelope

During the synthesis process, the AM function is ordinarily treated as though it is periodic; that is,

when the last segment of the function has been synthesized, the synthesis then restarts with the first segment again. However, if the Envelope option has been specified, the slope of the last segment of the AM function is maintained until a new signal is begun.

e. Music

When this option has been specified, the AM function is initialized for each signal during synthesis. This option is useful in the synthesis of musical sequences where it may be desired to have each signal represent a note of a melody, and to have the AM function simulate the attack and decay characteristics of a musical instrument. Using this option, it is unnecessary to specify an AM function for each signal during the signal definition process. All that is required is to specify the AM function for the first signal of the sequence. That function would be used and reinitialized for each signal during synthesis until a new AM function were specified in the sequence.

13. Comment

The purpose of this choice is to allow the user to record comments on the line printer, which is useful in making notes concerning the other material being printed. The Comment command may be used at three places in the program - when the list of primary operations is displayed, at the list of secondary choices obtained by entering the Print command, and at the option list obtained by entering the Execute command. After the Comment command has been

received, each line of text entered on the AGT typewriter is transferred to the line printer, with three exceptions. When the word TOP is entered as the first word of a line of text, the line printer will skip to the top of a new page; when STAR is entered, a row of asterisks is printed; when END is entered, the Comment operation is terminated.

14. Clear

The Clear command is used to erase the main signal sequence when deletion signal by signal is inconvenient.

15. Step

This choice is used to define a group of ten frequency elements, whose successive values increase by a fixed increment according to the following formula.

$$F_i = F_0 + i \cdot \Delta f, \quad i = 0, 1, 2, \dots, 9.$$

16. Octave

This choice is used to define an octave of frequency elements according to the following formula.

$$F_i = F_0 \cdot 2^{i/D}, \quad i = 0, 1, 2, \dots, D.$$

When the value of D is 12, an equally tempered musical scale results.

B. DATA STRUCTURE

The purpose of this section is to briefly outline the manner in which the digital computer program handles the data to be used by the analog computer synthesis.

1. Element Type Data

For each of the function elements (periodic wave, AM, and FM functions), two arrays of eight words each are

reserved. One of these arrays is used to describe the slopes of the eight segments of the function; the other is used to describe the eight breakpoints. The first 15 bits of each word specify the analog value to be transmitted to the analog computer; the last nine bits identify the channel over which the value will be transmitted. Since the digital to analog converter eventually requires the data to be converted to this format, processing during synthesis is reduced and computer core is conserved by storing it in this form initially. The data for the other signal elements is stored in a similar format, except for the duration elements, which are not converted to analog signals but are used in a digital form.

2. Main Signal Sequence Data

An array (ISIG) of 1000 words was reserved for the description of the main signal sequence. At the bottom of the array, each word corresponds to an individual signal and is used to partially describe that signal and to point to additional descriptive information. Words at the top of the array contain the additional descriptive information.

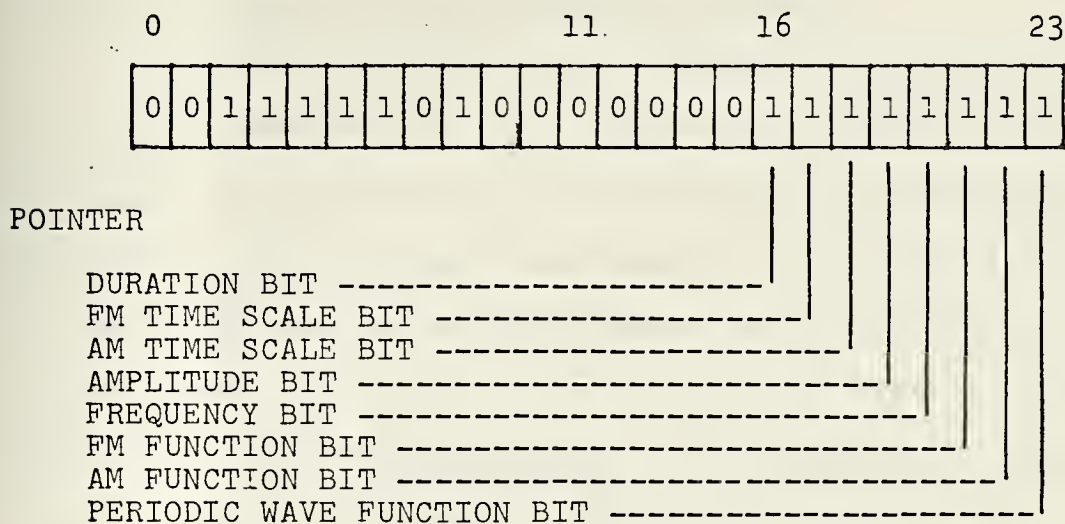
To illustrate how this is done, refer to Figure III-3 and to the following example. The first twelve bits of word number one are used as a pointer to the start of additional information about signal number one at the top of the array. In this case, the value of the pointer is $(1750)_8 = (1000)_{10}$. The last eight bits of word number one indicate which of the eight element types have been

FIGURE III-3

(a) Array ISIG Sample Data

1	(1000) ₁₀	(377) ₈	Bottom of the Array
2	(0992) ₁₀	(010) ₈	
3	(0991) ₁₀	(157) ₈	
992	(0022) ₁₀		Frequency Element #22
993	(3000) ₁₀		Duration = 3000 ms
994	(2800) ₁₀		FM Time Scale = 2800
995	(5000) ₁₀		AM Time Scale = 5000
996	(0007) ₁₀		Amplitude Element #7
997	(0041) ₁₀		Frequency Element #41
998	(0003) ₁₀		FM Function #3
999	(0001) ₁₀		AM Function #1
1000	(0005) ₁₀		Periodic Wave Function #5

(b) Binary Form of Word Number One



defined for signal number one. A one in bits 16 through 23 indicates that the element has been defined; a zero indicates that the corresponding element of the previous signal synthesized is to be continued. For signal number one, all eight elements have been defined. The number of one-bits in bits 16 through 23 also tells how many words at the top of the array are associated with the signal, with the pointer identifying the first of these words.

To continue the example, word number two points to the information at the top of the array describing signal number two. Only the frequency element of signal number two differs from the elements of signal number one. Thus, only a single bit of the last eight bits of word two is a one, and the pointer points to the word at the top of the array which identifies the new frequency (word 992).

From the description above, it can be seen that the maximum number of signals that the main sequence will contain is not fixed, but depends on the sum of the number of elements defined for the individual signals.

Just prior to the actual synthesis of signal number one, the information in word number one is unpacked and used to locate the description of the elements of the signal. When the description has been located, the actual analog values fully describing the elements are retrieved and are placed in special arrays to be used by the digital to analog converter and the data channel. To accomplish the actual conversion and transmission, these special arrays are

identified to the converter with special assembly language instructions [31].

While signal number one is being synthesized by the analog computer, word number two of ISIG is unpacked and processed in a similar manner. When signal number one ends, signal number two is converted and transmitted. Then signal number three is processed, etc. Additional information describing the synthesis process is contained in Section IV and in the computer program.

IV. ANALOG COMPUTER CIRCUITS

A. FUNCTION SEGMENT SLOPE SWITCHING

Refer to Figure IV-1. The eight segment slope values of the normalized periodic wave function are transmitted by the digital program to terminals T420 - T427. Each slope value is attenuated (by 50%) by a potentiometer to reduce switching transients and is then fed to a digital-analog switch. The top four switching components shown are COMCOR amplifiers specifically designed for switching; the bottom four switches are combination summer-integrators used as switches by patching the inputs to the IC (initial condition) terminals of the integrators and deleting the integrator feedback capacitors. Switching is effected by means of the reset-compute mode controls. An end-around shift register (Figure IV-8) is used to switch in one segment at a time to generate the function. The normalized function segment slope values may be further attenuated by potentiometer P407 before being sent to M003 where they are multiplied by the composite frequency signal (see Figure IV-4).

B. FUNCTION SEGMENT BREAKPOINT STORAGE AND SWITCHING

The breakpoint values of the segments of the normalized periodic wave function are transmitted to terminals T420 - T426 by the digital program and are attenuated (by 50%) by potentiometers (see Figure IV-1) and then fed to track-hold networks (refer to Figure IV-2). The top four track-holds

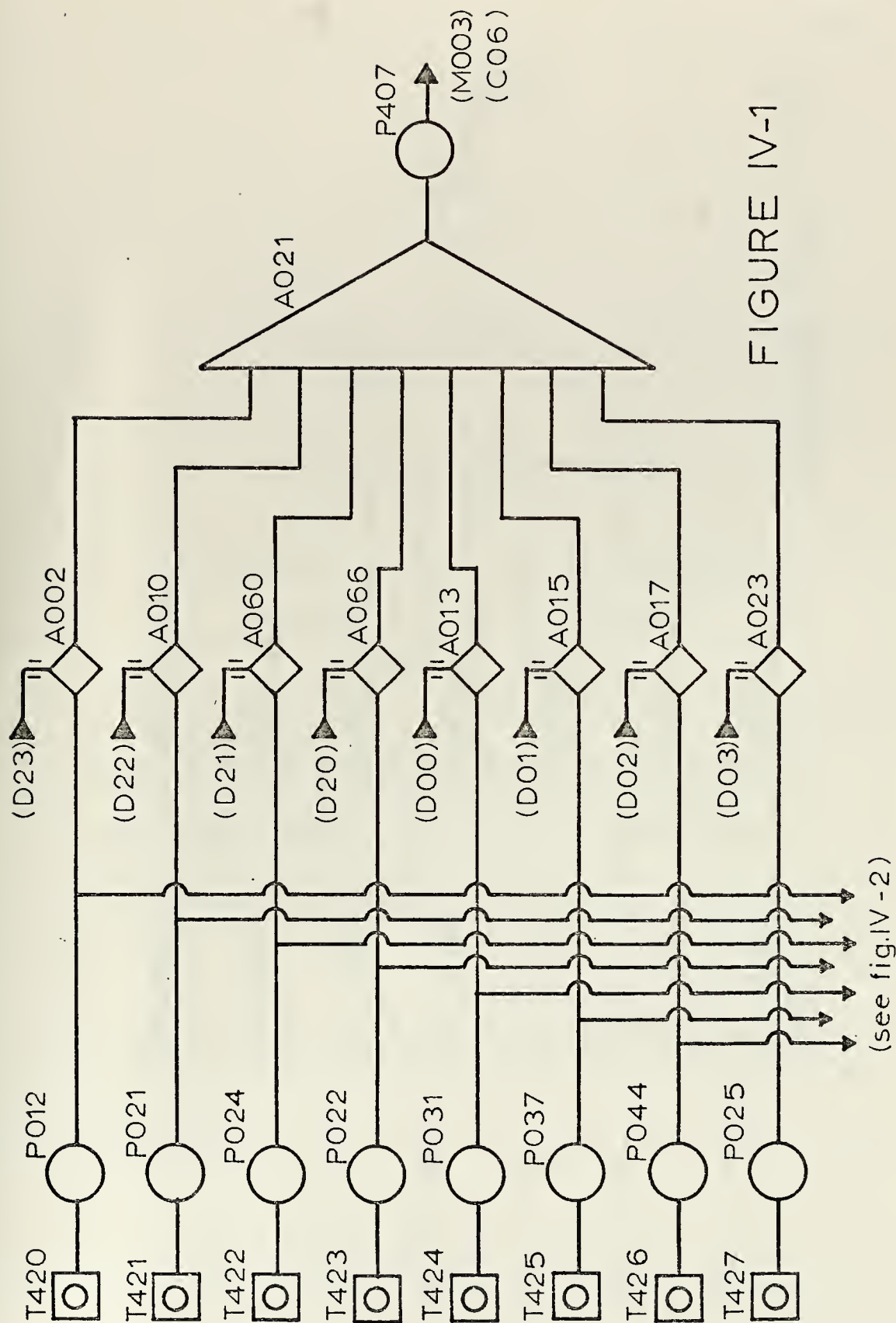


FIGURE IV-1

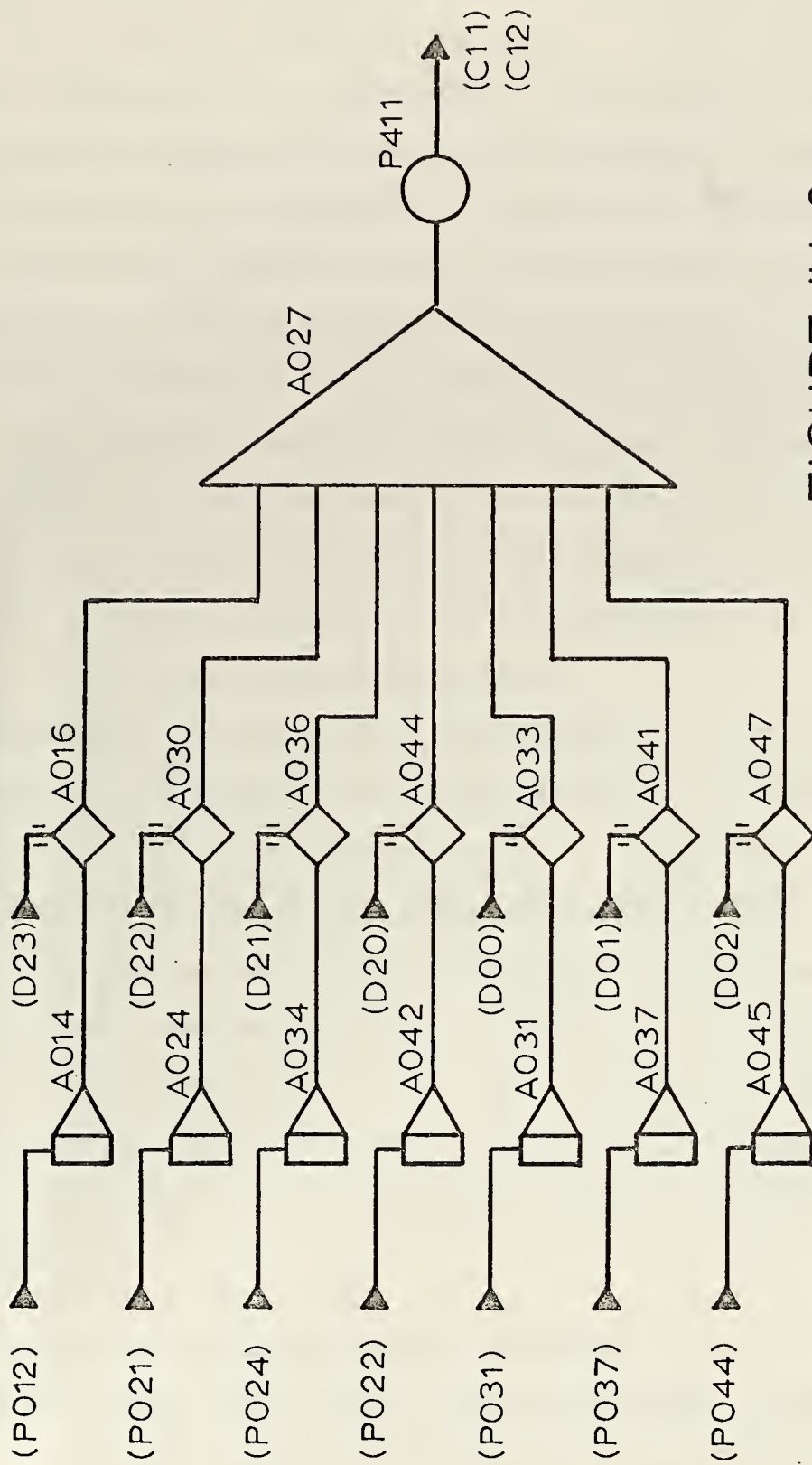


FIGURE IV-2

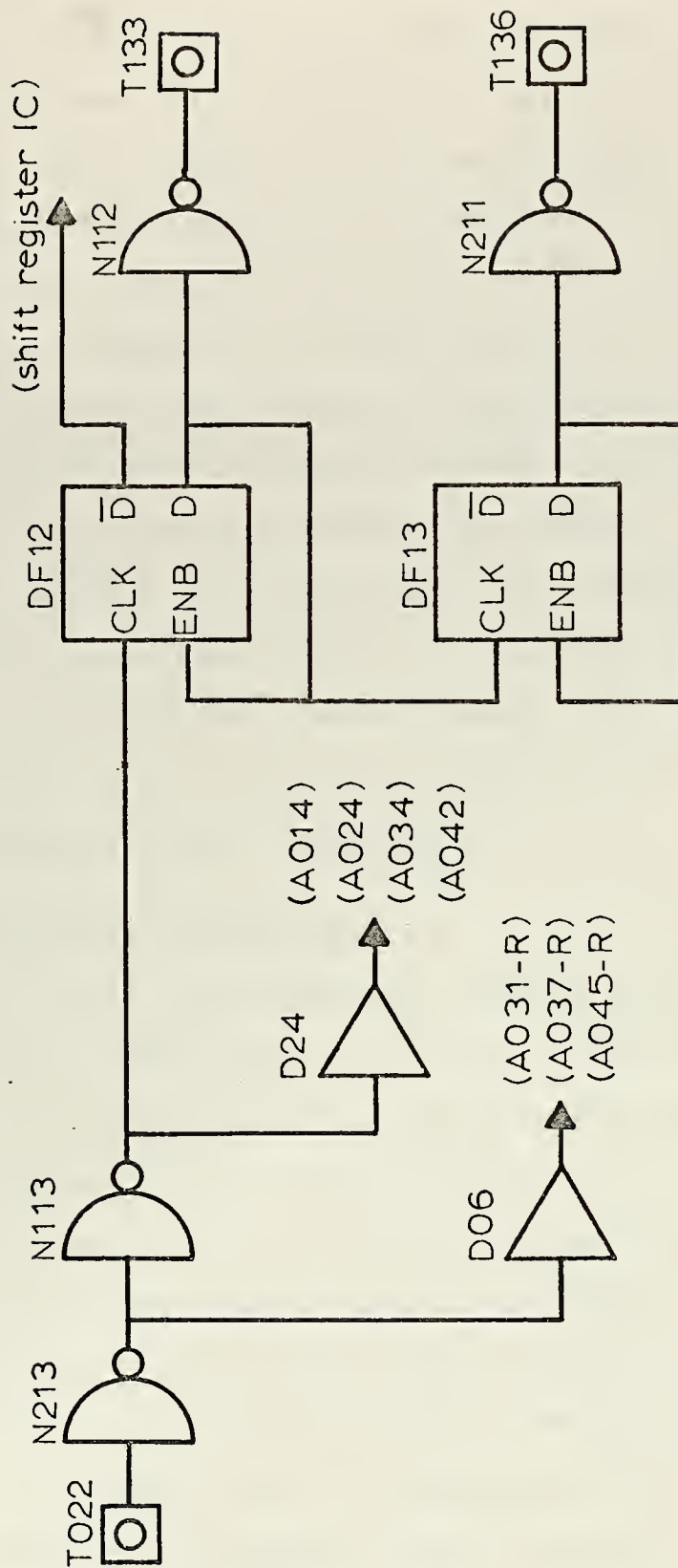
shown in Figure IV-2 are COMCOR components specifically designed for that purpose; the bottom three are integrators used as track-holds by patching the inputs to the initial condition terminals of the integrators. Both types of components function equally well but require different logic inputs to perform the track and hold functions. The breakpoint of the eighth segment is zero and requires no input. The breakpoint values are tracked for approximately 500 microseconds and are then held. Storing the breakpoint values with track-hold networks allows the same terminals (T420 - T427) and potentiometers to be used for both the slopes and the breakpoints of the function segments. This economy was necessary because of the limited number of channels in the digital-analog converter.

The breakpoint values are switched into the function generator circuit (Figure IV-4) one at a time by the digital-analog switches, which are controlled by the end-around shift register mentioned in the previous section. The breakpoint values can be further attenuated by potentiometer P411, which functions as a time scale adjustment. Increasing the amount of attenuation increases the frequency of the generated function and decreases its magnitude. The output of P411 is then applied to the function generator circuit (Figure IV-4).

C. PERIODIC WAVE FUNCTION GENERATOR CONTROL LOGIC

Refer to Figure IV-3. Before transmitting the breakpoints of the periodic wave function, the digital program transmits

FIGURE IV-3



a zero logic signal to T022. This puts the track-hold networks (A014, A024, A034, A042, A031, A037, and A045) in the track mode and causes DF12 to generate a pulse to reset the shift register. A031, A037, and A045 are combination summer-integrators used as track-holds, which accounts for the difference in logic signal between them and A014, A024, A034, and A042. The breakpoints are then transmitted. The pulse generated by DF12 is long enough to insure that the track-holds have tracked the breakpoint values. The program tests T133 to determine when the tracking is complete. When it is, the program tests T136 to determine when the track-holds have switched to the hold mode; DF13 generates a short pulse to allow the transition to the hold mode. When the hold mode is detected, the slopes of the segments of the periodic wave function are then transmitted.

D. PERIODIC WAVE FUNCTION GENERATOR

Refer to Figure IV-4. The normalized segment slope values of the periodic wave function are fed to multiplier M003 where they are multiplied by the composite frequency signal from A011 (Figure IV-12). A025 integrates the output of M003 to generate the segment. Unlike the FM and AM function generators, the periodic wave function generator compares the "X" (or time) value of the segment with the corresponding breakpoint value to determine when the segment breakpoint has been reached. This is accomplished by integrators A057 and A061, which integrate the frequency signal

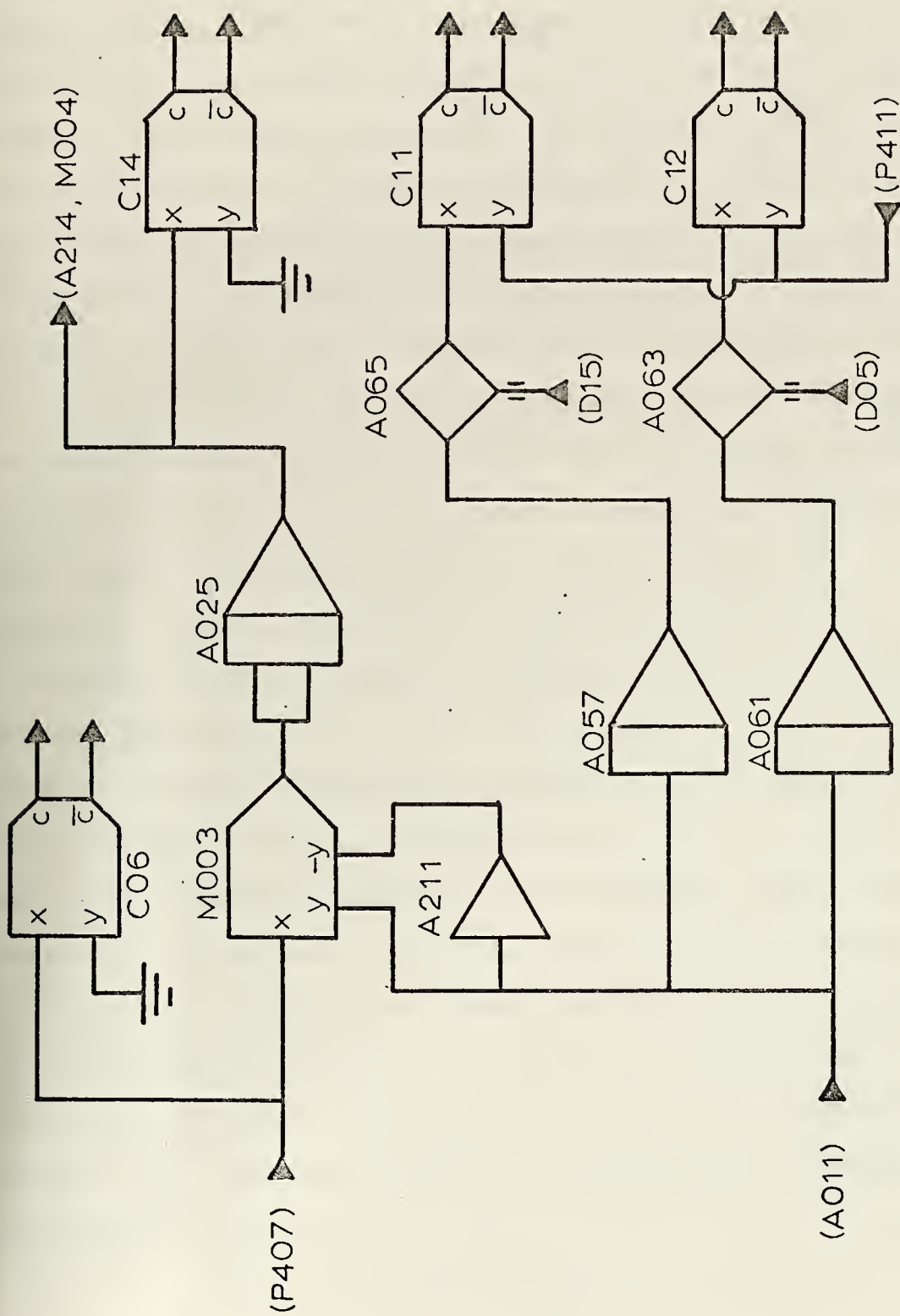


FIGURE IV - 4

to generate a sawtooth timing voltage, whose value is proportional to the "X" value of the periodic wave function; the period of the timing voltage is determined by the frequency signal. Only one of the timing signal integrators is used at a time, each integrator being used on alternate cycles. This toggle arrangement was necessary because of the time required to reset an integrator at the end of a cycle (one integrator is being reset while the other is being used). Switches A063 and A065 are used to isolate the integrator being reset. For the first seven segments of the function, the timing voltage is compared with the breakpoint values in either comparator C11 or C12 to determine when the breakpoint has been reached. When a breakpoint is detected, the values of the next segment are switched into the generator circuit and the old values are disconnected.

This technique of using a separately generated timing voltage provides no positive control over the output voltage of A025; without some control, a small bias or noise at the input of A025 would cause the dc value of the output to drift over a number of cycles, and eventually A025 would overload. Therefore, to provide control over the output of A025, the "Y" value, rather than the "X" value of the eighth breakpoint of the function is used for breakpoint detection. Since the "Y" value of the eighth, or last, breakpoint is always zero, each wave period is automatically compensated for any dc drift.

E. TIMING SIGNAL GENERATOR LOGIC

Refer to Figure IV-5. Each time the slope and breakpoint values of the first segment of the periodic wave function are switched into the function generator circuit, the input to N124 (from the shift register of Figure IV-8) causes the outputs of D05 and D15 to reverse states. As long as the circuit is not disabled by a zero input from D04, D05 and D15 are always in opposite states. The circuit is used to switch between the timing voltage generators of Figure IV-4.

F. PERIODIC WAVE FUNCTION GENERATOR BREAKPOINT DETECTOR

Refer to Figure IV-6. For the first seven segments of the function, the function timing signal is compared with the negative of the segment breakpoints in comparators C11 or C12 (the upper half of the circuit is disabled by a zero input from FF08). When a breakpoint is reached, the output of N422 changes from a logical one to a logical zero; the same change of state occurs at the output of N127. The output of N127 is fed to the pulse generator circuit shown in Figure IV-7.

For the eighth segment of the function, the lower half of the circuit is disabled by a zero signal from FF08' (other zero inputs to N422 will also disable this part of the circuit) and the upper half of the circuit is enabled. The upper half (C06, C14, N207, N212, and N412) is used to determine when the function value reaches zero (the

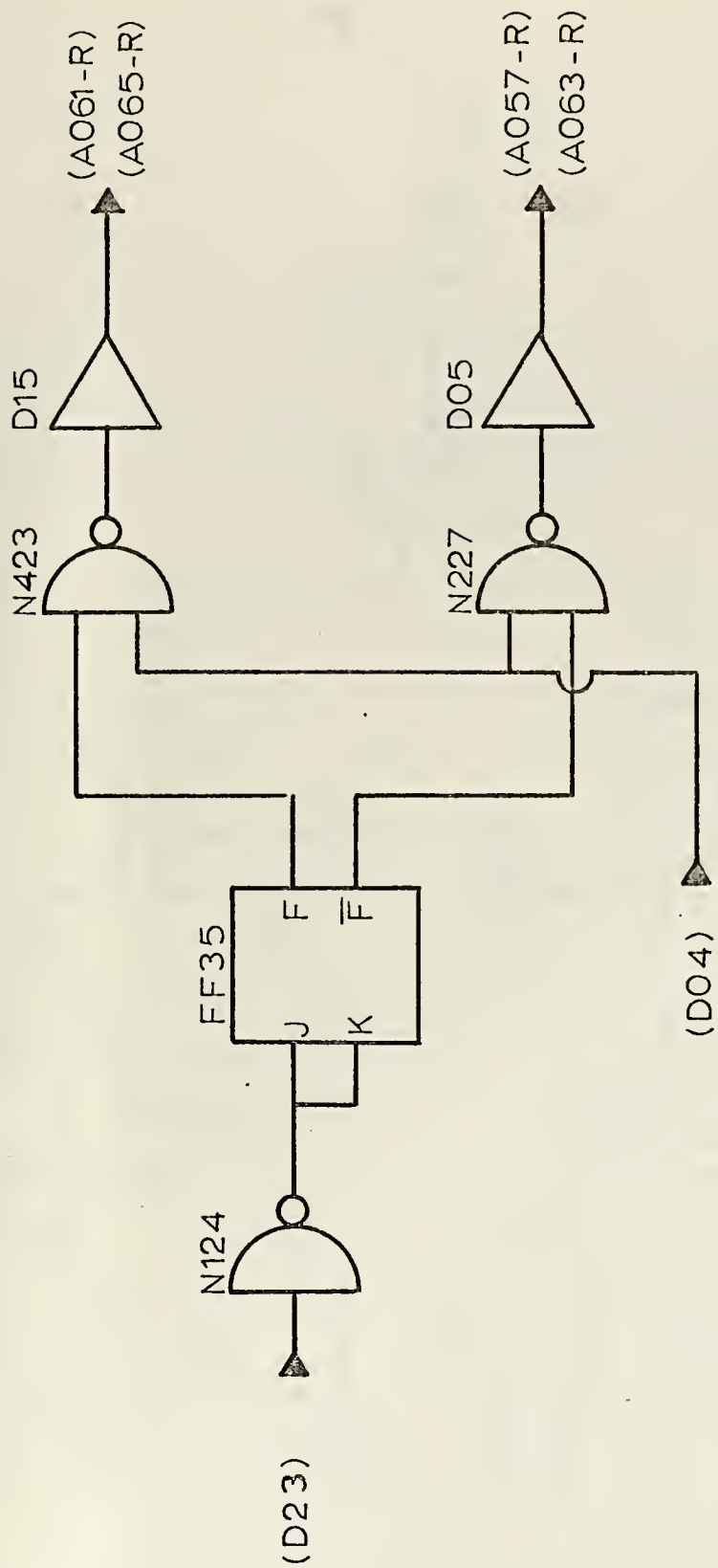


FIGURE IV-5

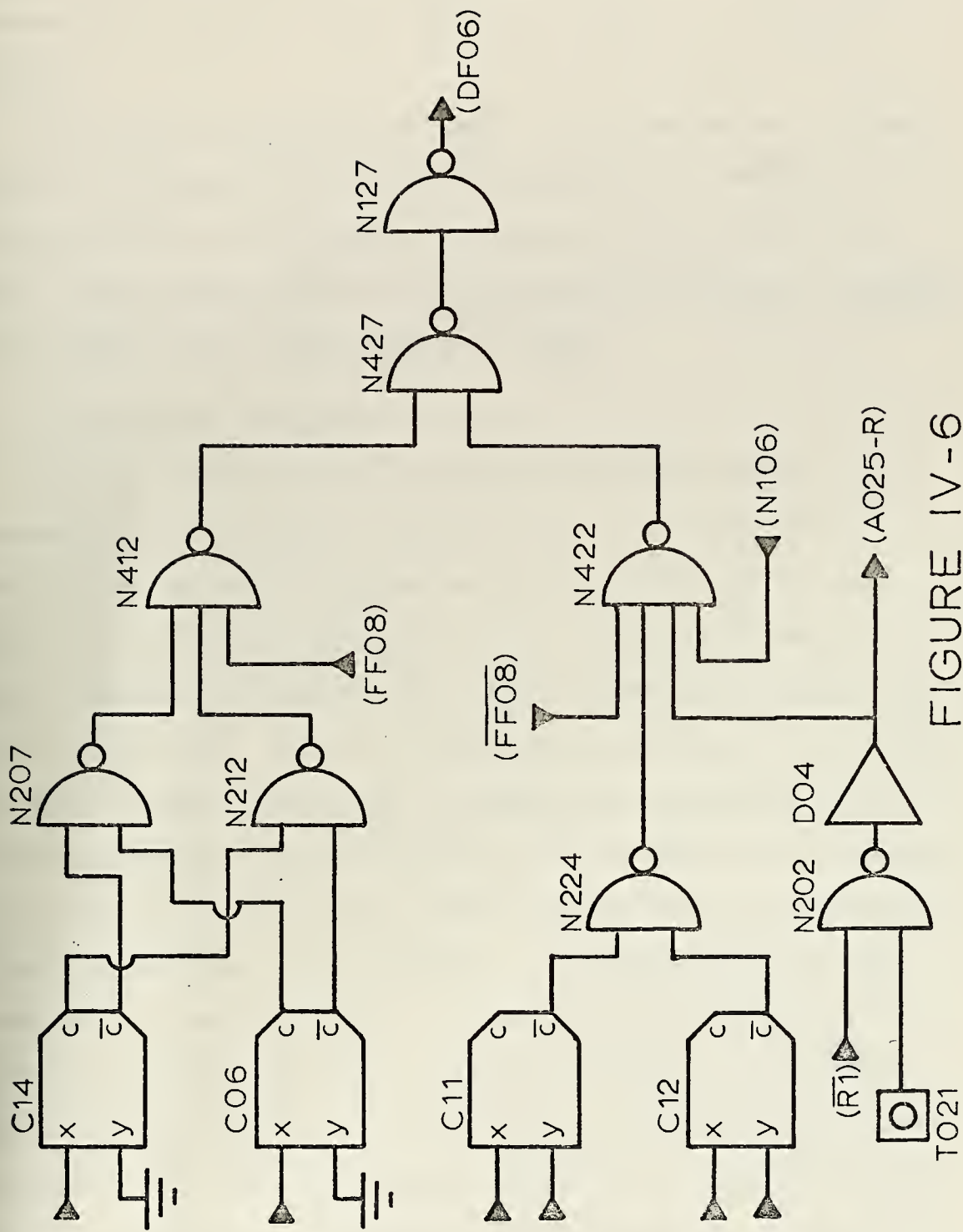


FIGURE IV-6

breakpoint of the eighth segment). Its operation is described in the section explaining the operation of the breakpoint detector of the AM function generator (Figure IV-10).

When the analog computer is in the reset mode, a signal from R1' (through N202 and D04) disables the breakpoint detector circuit and places integrator A025 in the reset mode. The same thing is accomplished by the digital program by sending a zero logic signal to T021.

G. SHIFT PULSE GENERATOR

Refer to Figure IV-7. Each time the breakpoint of a segment of the periodic wave function is reached, the breakpoint detector triggers delay-flop DF06, which generates a 15 microsecond pulse that is fed to the segment shift register (Figure IV-8). The circuit also serves two other functions. First, it feeds back a signal (from N106) to the breakpoint detector to disable it during switching from one segment to the next and long enough for the detector comparators to recover. This was necessary to prevent false triggering of the detector. Experimentally it was determined that a minimum of about 65 microseconds was required for this. Second, an additional delay is required to allow the delay-flops to recover; a minimum of 10 microseconds delay after the output of the delay-flop returns to zero is necessary. To provide these delays and to prevent false triggering, three delay-flops were used. Their timing

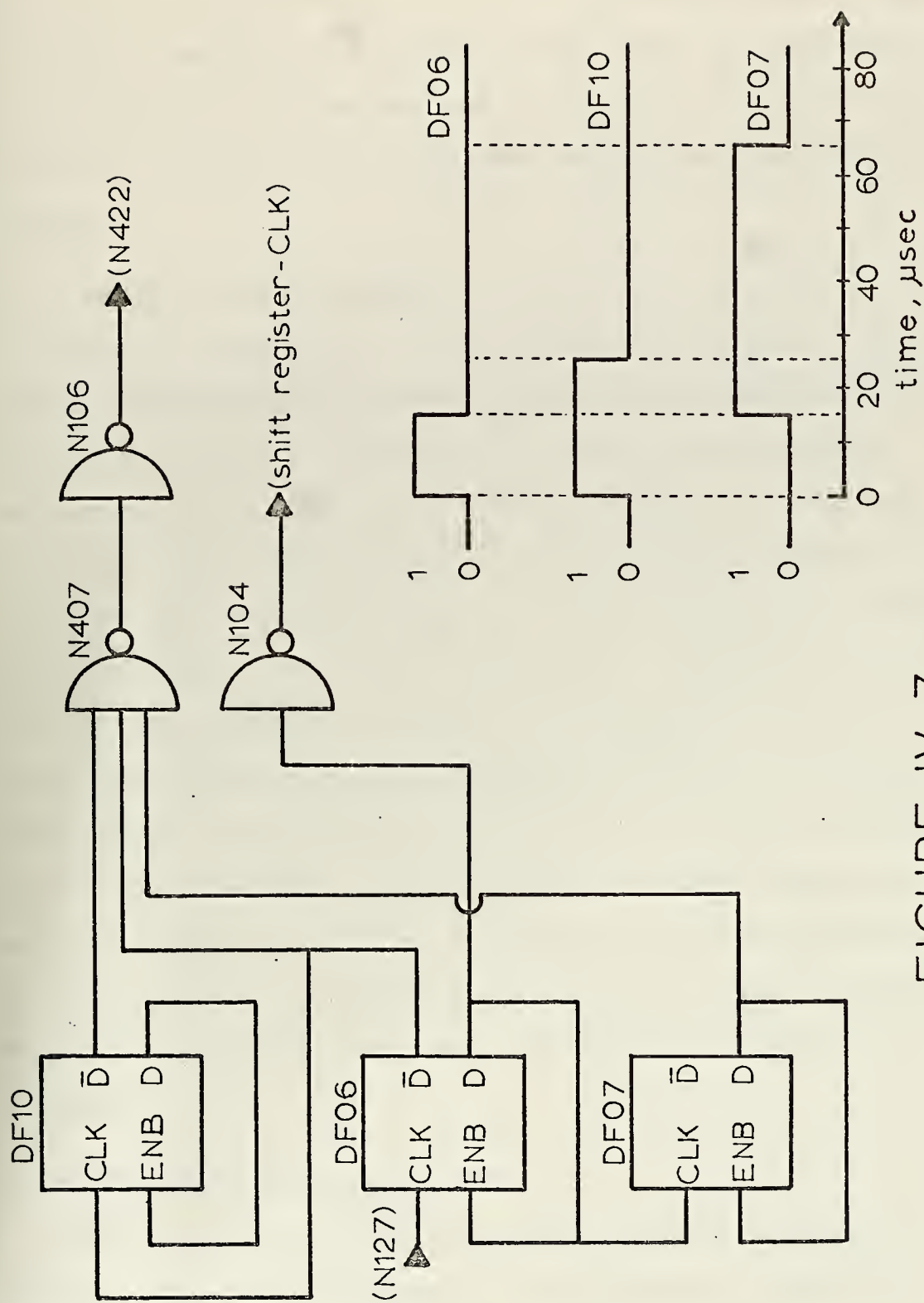


FIGURE IV-7

is shown in Figure IV-7. The pulse from DF10 overlaps that of DF06 to maintain a continuous zero output at N106 for the minimum 65 microseconds and is necessary because of the transient that may occur when DF06 and DF07 change state together.

H. SEGMENT SHIFT REGISTER

Refer to Figure IV-8. The periodic wave function segment shift register is used to control the connection of the segment slope and breakpoint values to the function generator circuit. A signal from DF12 initially sets the first bit of the shift register and resets all others. With only the first bit "on," only the values of the first segment are connected to the function generator. When the breakpoint of the first segment is reached, the breakpoint detector and pulse generator circuits (Figures IV-6 and IV-7) generate a pulse that shifts the "on" bit to the second position in the shift register, disconnecting the first segment values and connecting the second. The output of the last position of the register is returned to the register input to provide end-around shifting that makes the generated function periodic.

I. AM FUNCTION GENERATOR

Refer to Figure IV-9. The analog signals present at T431 and T407 are the slopes of the alternate segments of the normalized AM function. At T430 and T406 are the corresponding values of the breakpoints. The initial value

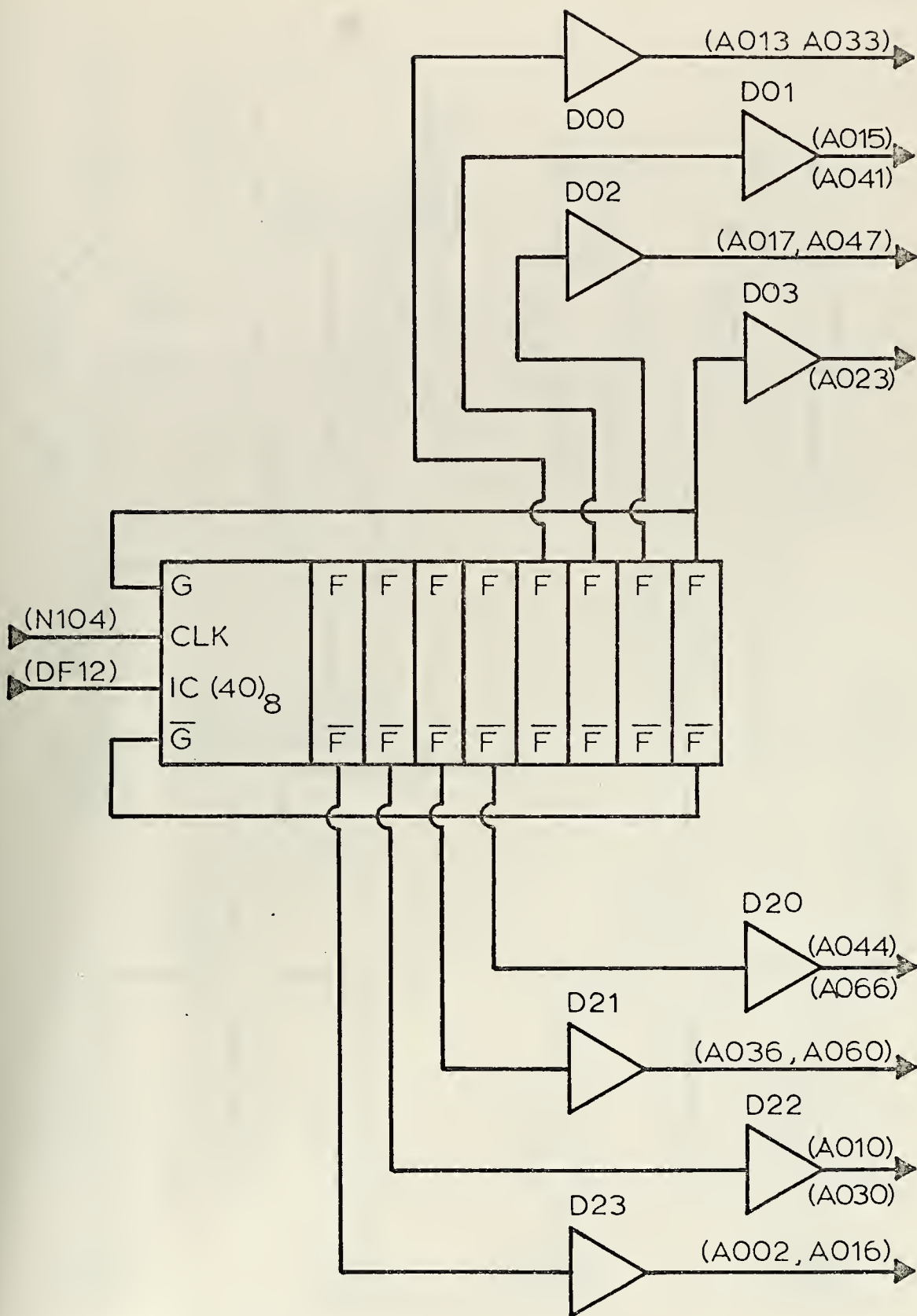


FIGURE IV-8

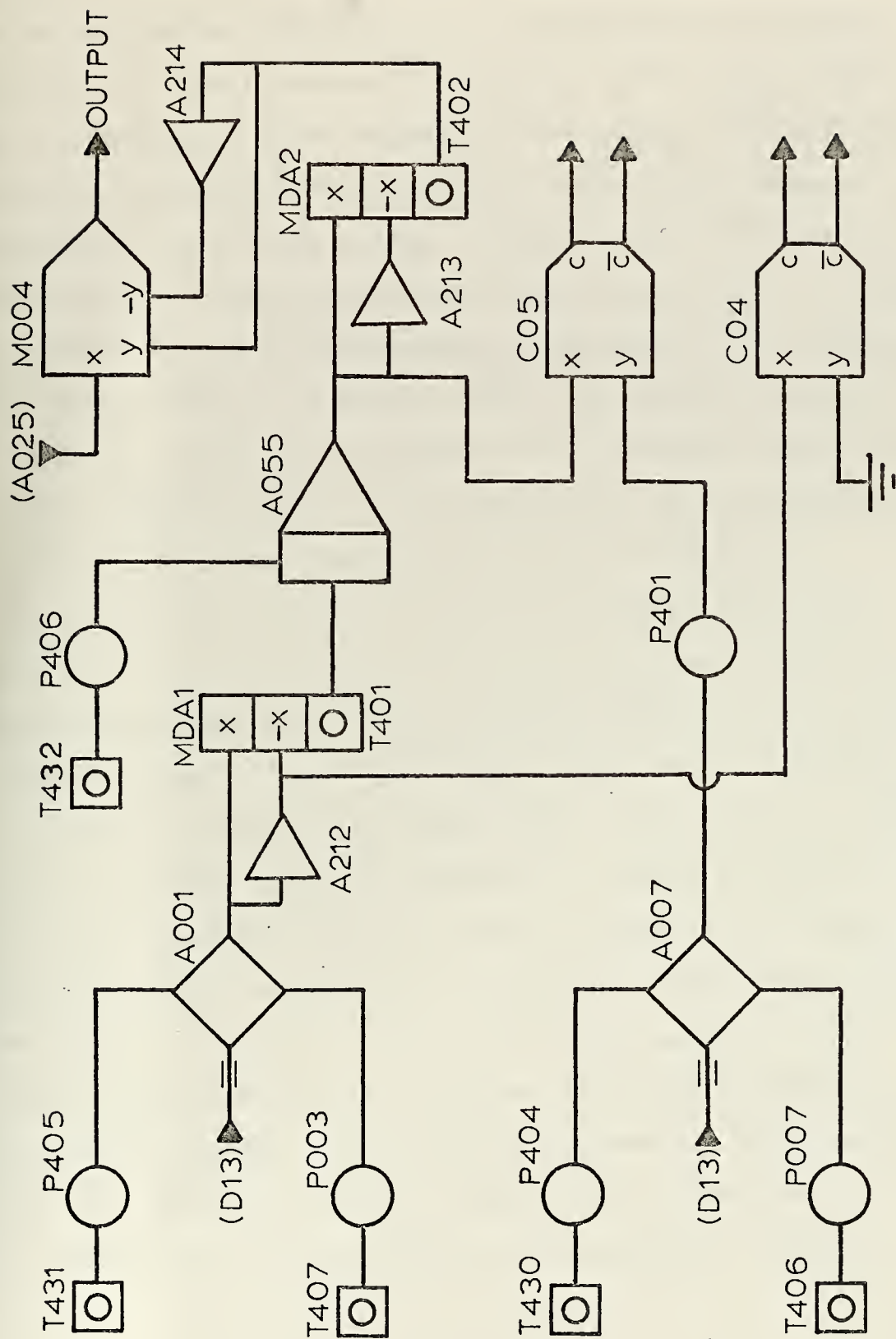


FIGURE IV-9

of the function appears at T432. Potentiometers are used to reduce analog voltages to one half of their original values to reduce undesirable transient effects that occur when switching from one segment to the next. A001 and A007 function as toggle switches to select the proper segment, depending on the logic signal from D13. These two-position switches were created by using summer-integrators. One input is patched to the initial condition terminal of the integrator and the other to a normal input; the feedback capacitor is deleted. When the integrator is in the reset mode, the initial condition input is selected; when the integrator is in the compute mode, the other input is selected.

The normalized slope value of the segment being generated is multiplied by the AM function time scale element in multiplying digital to analog converter MDA1 and the product appears at T401. The slope value is then fed to integrator A055 which generates the segment. The segment value, breakpoint, and slope sign are directed to comparators C04 and C05, which together with the other AM function generator logic components detect when the segment value reaches the breakpoint. When this occurs, the logic signal from D13 changes state and switches A001 and A007 from the (N)th to the (N+1)th segment. While the new segment is being generated, the values of the next segment are transmitted to the unused poles of the switches by the digital program.

The time scale of the AM function may also be adjusted by P401. Decreasing the setting of P401 has the effect of decreasing the period and the amplitude of the function.

The generated function appearing at the output of A055 is then multiplied by the amplitude element in MDA2 and the product appears at T402. From there it is directed to multiplier M004, where it is multiplied by the periodic wave function from A025. The output of M004 is the final modulated signal.

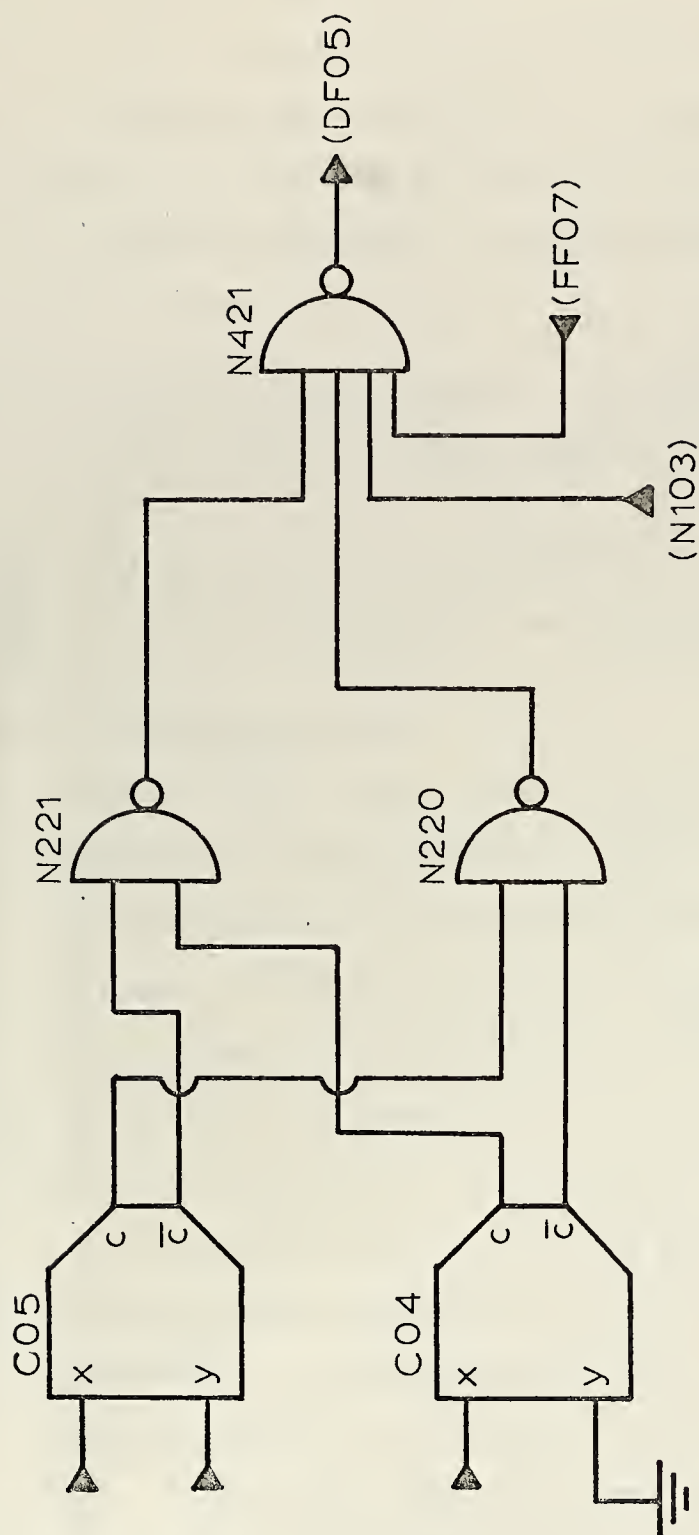
J. AM FUNCTION GENERATOR BREAKPOINT DETECTOR

Refer to Figure IV-10. This circuit is used to detect when the value of a segment being generated reaches the breakpoint of that segment. Since the functions being generated are not monotonic, this can occur under one of four possible conditions - with the slope positive or negative and the breakpoint positive or negative. The states of the outputs of the comparators C04 and C05, before and after the breakpoint is reached, for each of the four possible conditions is shown in the table accompanying Figure IV-10. The output required from the circuit was a change of state from a logical one to a logical zero when the breakpoint was reached. From the state table, it can be seen that the appropriate equations for the output are

$$N421' = C4'C5' + C4 C5 \quad \text{and} \quad N421 = C4 C5' + C4'C5$$

which can be equivalently expressed as

FIGURE IV-10



SLOPE	-BREAKPOINT	$c04^-$	$c04^+$	$c05^-$	$c05^+$
-	-	1	0	0	0
-	+	1	0	0	0
+	-	0	1	1	1
+	+	0	1	1	1

$$N421 = ((C4 \ C5)')' (C4' \ C5)')' \ .$$

The first term on the right hand side of the equation is the output of N221 and the second term is the output of N220. The inputs to N421 from N103 and FF07 are used to disable the breakpoint detector. Including them, the equation for the output of N421 is

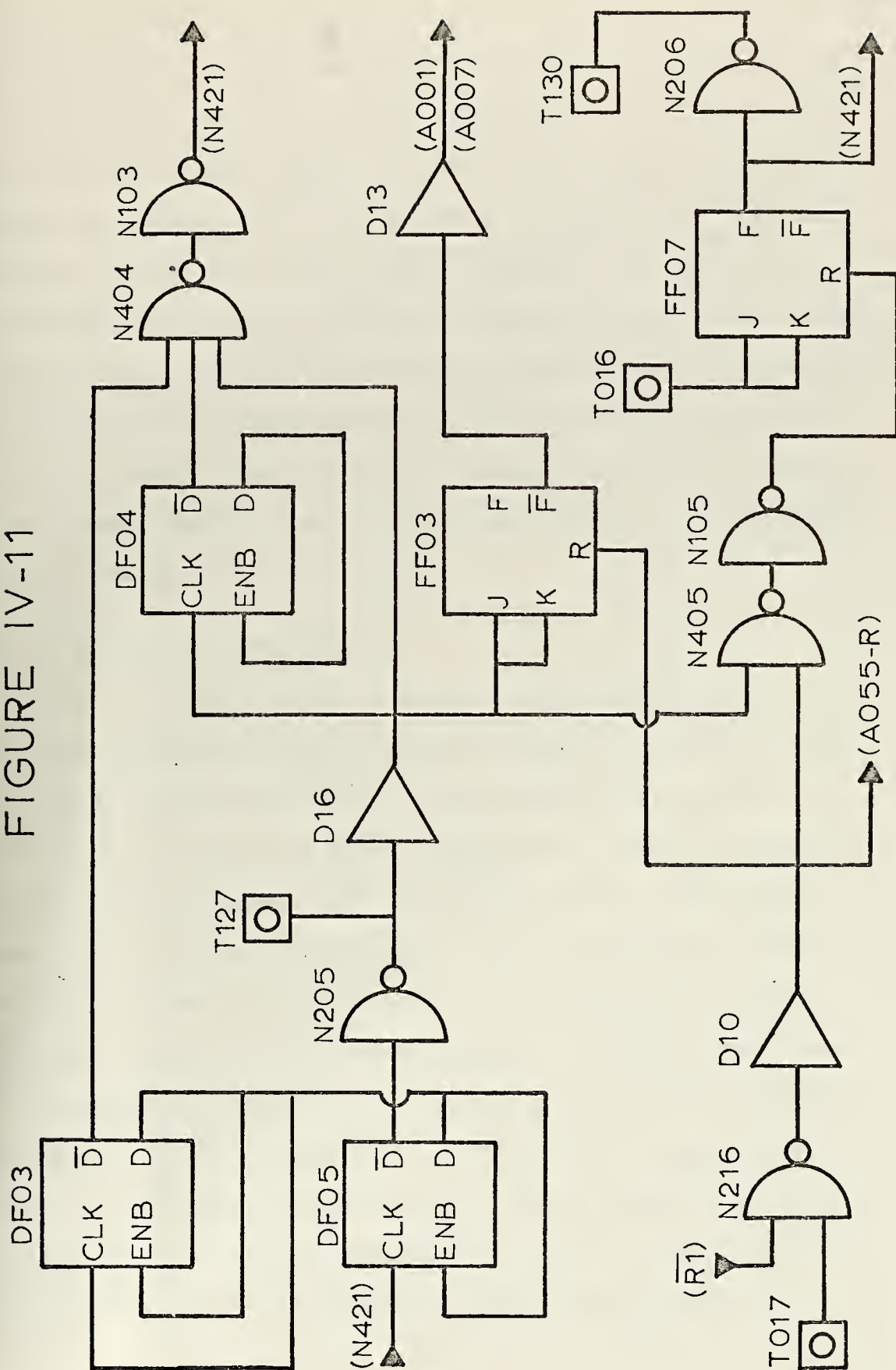
$$\begin{aligned} N421 &= (N220 \ N221 \ N103 \ FF07)' \\ &= N220' + N221' + N103' + FF07' \end{aligned}$$

This circuit configuration is also utilized in the FM function generator and the periodic wave function generator.

K. AM FUNCTION GENERATOR LOGIC

Refer to Figure IV-11. The output of the breakpoint detector appears as an input to DF05. DF05, DF04, and DF03 function in the same manner as the pulse generator circuit of the periodic wave function generator (Figure IV-3). Delay-flop DF05 generates a 25 microsecond pulse that is fed through N205, N405, D16, and N105 to reset FF07. The output of FF07 is fed back to N421 to disable the breakpoint detector; it is also fed through buffering gate N206 to T130 where the digital computer program tests the state of the flip-flop to determine if the breakpoint has been reached. The output of D16 appears as an input to FF03, causing it to change state. This causes A001 and A007 to switch to the new segment values (see Figure IV-9). After the digital

FIGURE IV-11



computer program has determined that the breakpoint has been reached, it tests terminal T127 to determine when the 25 microsecond pulse from DF05 is complete. When it is, the digital program sends a pulse to T016 to set FF07, which rearms the breakpoint detector when the output of N103 has returned to the one state. When changing from one function to another, the digital program transmits a zero logic signal to T017, which causes the breakpoint detector to be disabled by resetting FF07 and which also places integrator A055 in the reset mode. When the analog computer is put in the master reset mode, a zero logic signal from R1' accomplishes the same thing.

L. FM FUNCTION GENERATOR

Refer to Figure IV-12. Segment slope values appear at T433 and breakpoint values appear at T404. Both are attenuated 50% by potentiometers. Track-hold networks A050 and A064 hold the values of the (N)th segment while the values of the (N+1)th segment wait at T433 and T404. The slope value of the segment being generated is multiplied by the FM function time scale in multiplying digital to analog converter MDA0 and the product appears at T400. The slope is then fed to integrator A053 which generates the segment. The segment value, breakpoint, and the slope sign are directed to comparators C01 and C02, which together with the rest of the FM function generator logic are used to detect when the breakpoint is reached. When this occurs, the

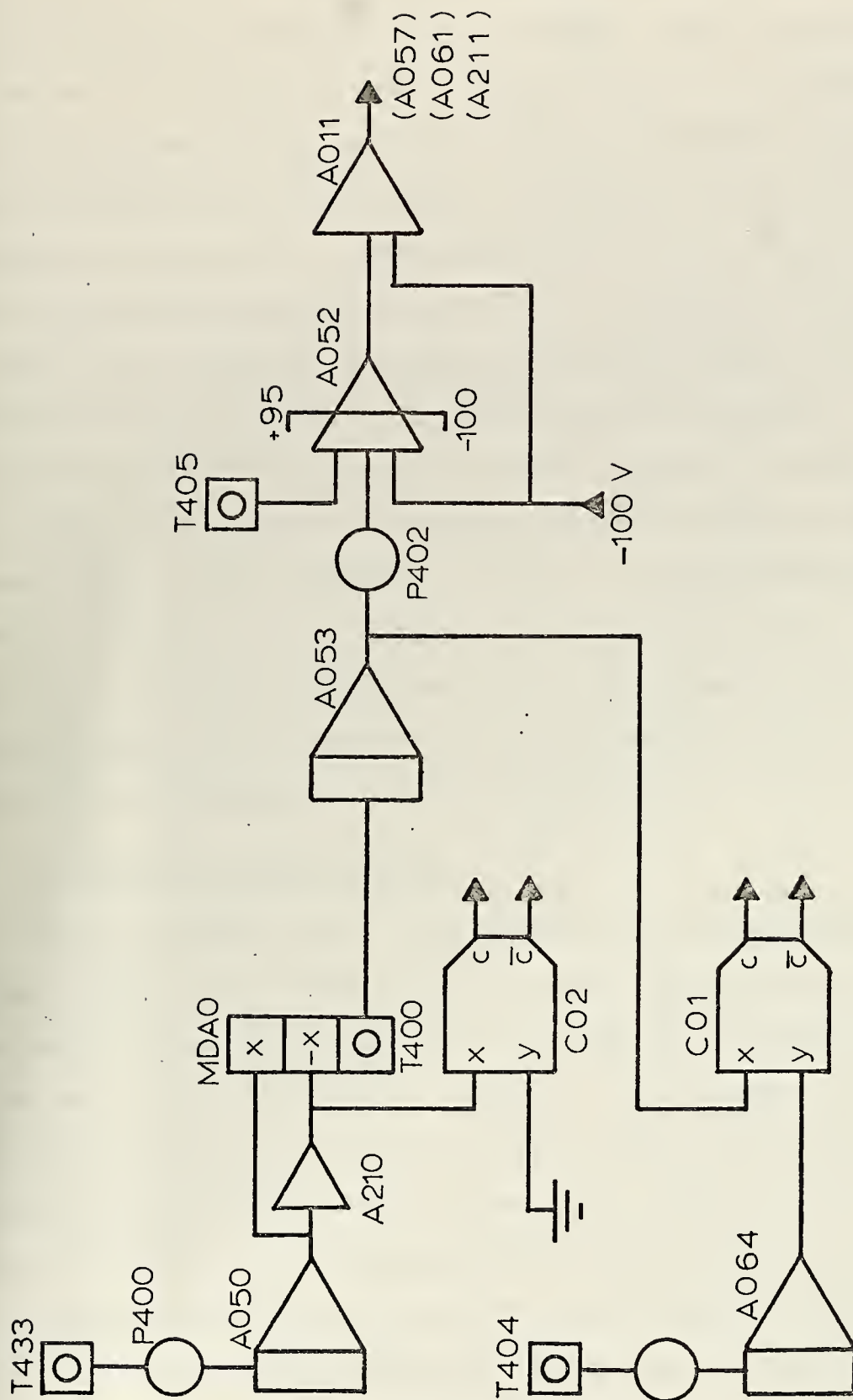
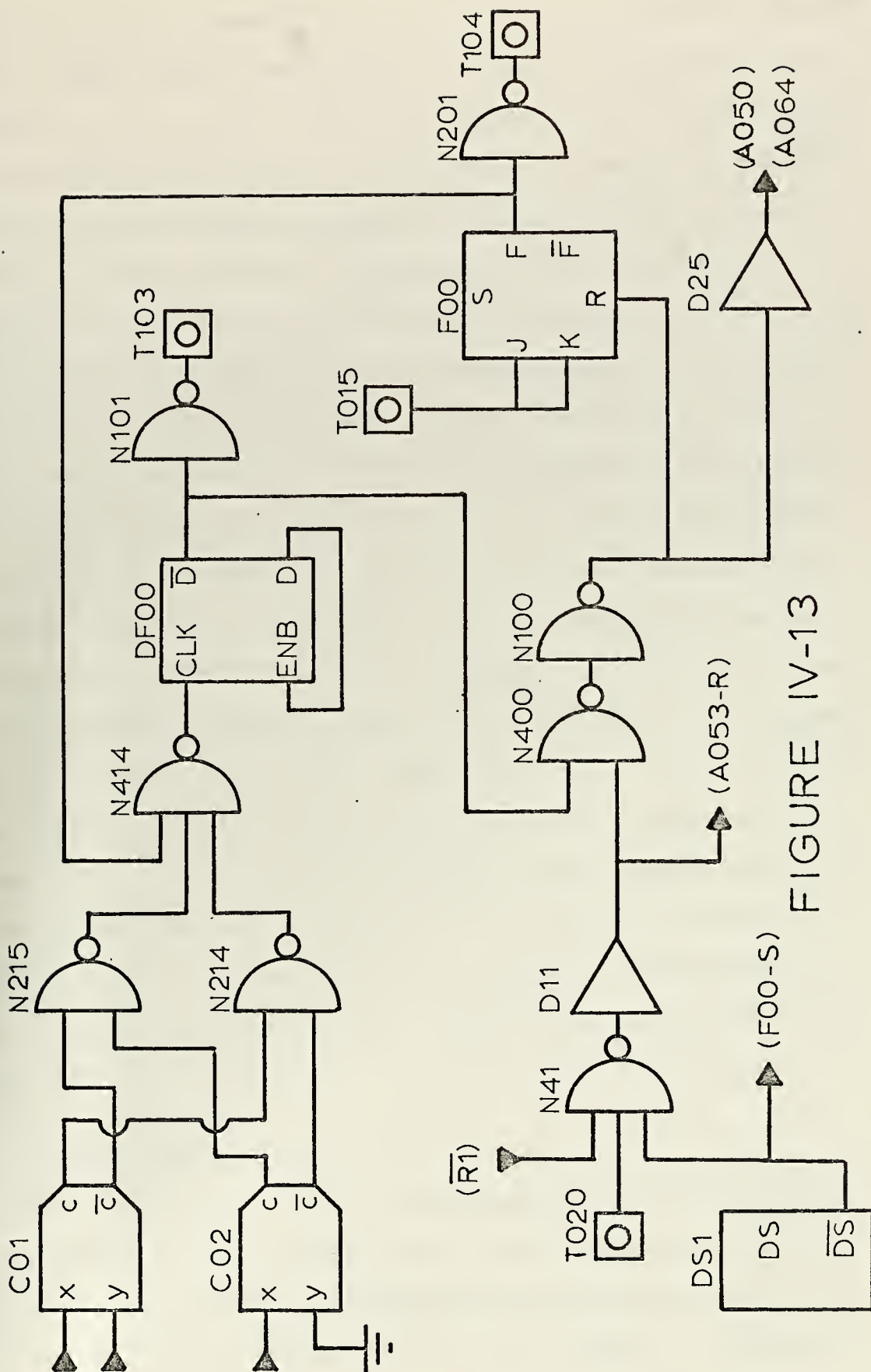


FIGURE IV-12

track-holds track the new segment for approximately 500 microseconds and then hold the new values. After the track-holds return to the hold mode, values for the next segment are transmitted to T433 and T404. After the eighth segment has been generated, the process is repeated. The generated function is directed to summer A052 where it is combined with the signal frequency element, which has been transmitted to T405. This combined frequency function is directed through amplifier A011 to the periodic wave function generator. Amplifiers A011 and A052 serve several functions. First, A052 limits the combined frequency function voltage to the linear range of the amplifier. Second, by setting the upper limit of A052 to a value slightly less than 100 volts, the combined frequency signal appearing at the output of A011 is always kept greater than zero (a requirement imposed by the periodic wave function generator circuit).

M. FM FUNCTION GENERATOR LOGIC

Refer to Figure IV-13. The operation of the breakpoint detector (C01, C02, N214, N215, and N414) is described in the section covering the AM function generator breakpoint detector. When a breakpoint is reached, the change of state from one to zero at the input of DF00 causes the delay-flop to generate a 500 microsecond pulse. The output of the delay-flop (D') is fed through N400 and N100, where it appears as a reset pulse for flip-flop FF00. The output of FF00 is fed back to N414 to disable the breakpoint detector when



the flip-flop is reset. This action is necessary to prevent false triggering of the delay flop when switching from one segment to the next. The output of FF00 is also fed through a buffering gate, N201, to T104, where its state is tested by the digital computer program to determine if the breakpoint has been reached. The output of N100 is also fed through driver D25 to the track-hold networks, A050 and A064, causing them to track the new segment values for 500 microseconds and then hold the new values. The status of the track-holds is tested by the digital program at T103 to determine when the new segment values have been tracked and are being held. When the program determines that the new segment values are being held, the next segment values are transmitted to T404 and T433 (see Figure IV-12) and the program transmits a pulse to T105 to set FF00 and arm the breakpoint detector. Zero logic inputs to N417 are used to disarm the breakpoint detector and place integrator A053 in the reset mode. This action is required when changing from one function to another, at which time the program transmits a zero to T020. Placing the analog computer in the master reset mode or using the digital switch DS1 accomplished the same thing.

N. SIGNAL TIMER

Refer to Figure IV-14. Flip-flops FF10-13 and FF20-27 are patched to form a binary down counter (the duration counter) and are used to time the signal being synthesized. The initial states of the first 14 flip-flops are used to

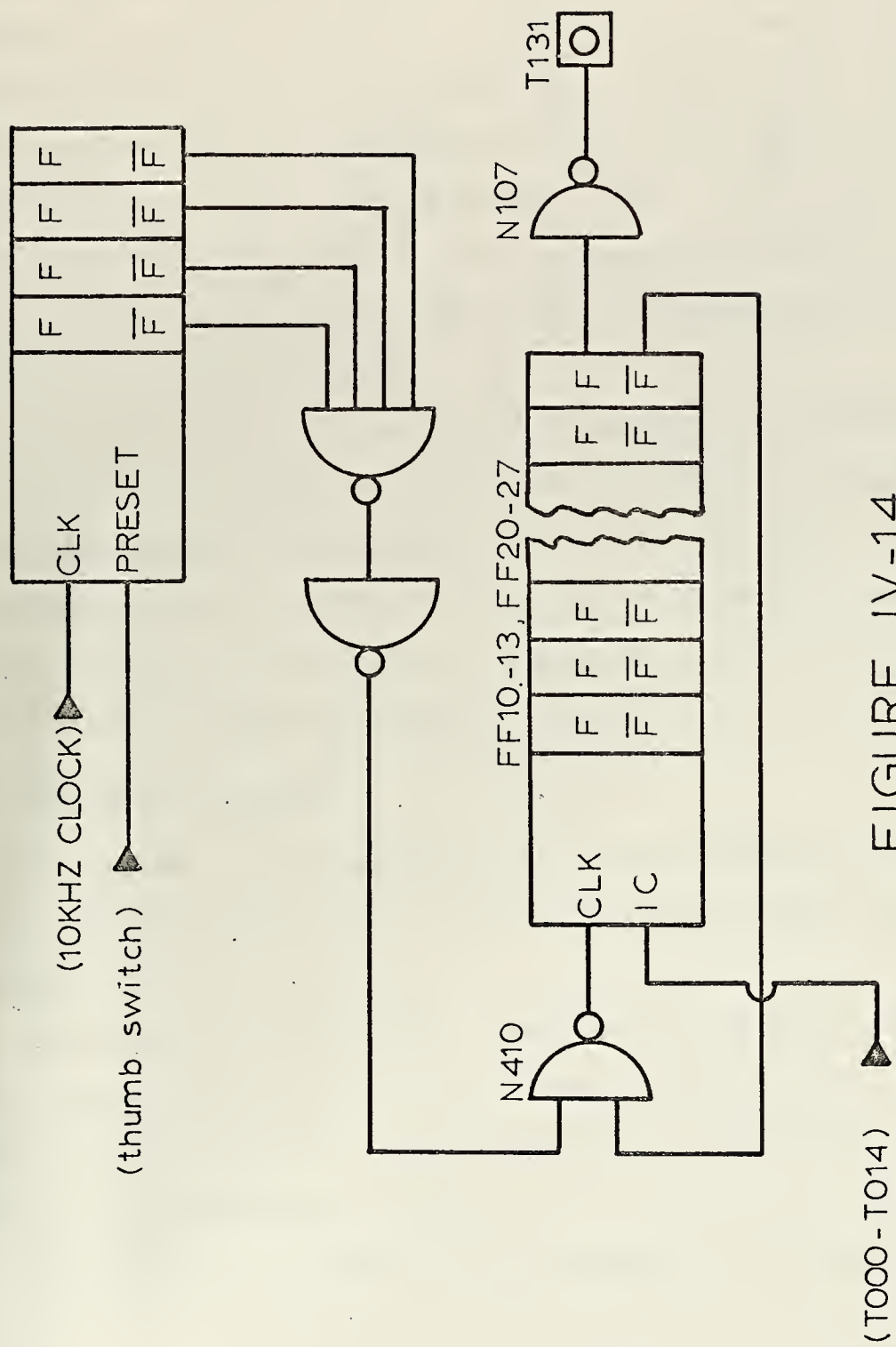


FIGURE IV-14

represent the binary length of the signal duration, with the fourteenth corresponding to the 2^0 bit. They are initialized by pulses from T001 - T014. The fifteenth bit is initially reset by a pulse from T000. Clock pulses to N410 cause the counter to count down, and when the first fourteen bits reach zero, FF27 changes state. FF27' is fed back to N410 to disable the counter. This condition is sensed at T131 by the program to determine that the signal has run for its prescribed duration.

The four-bit preset down counter (at the top of Figure IV-14), fed by a 10kHz clock, is used to generate the clock pulse that drives the duration counter through N410. By changing the preset condition with the thumb switch, frequency division of the 10 kHz clock pulse can be accomplished, thereby time scaling the duration counter.

O. COMPONENT SETTINGS

The inputs to all amplifiers use a gain setting of one except the following, which use a gain of ten: A025, A057, and A061.

Feedback capacitors, where used, are as follows:

A025 - 0.001 μ F, A031 - 0.010 μ F, A037 - 0.010 μ F,
A045 - 0.010 μ F, A053 - 0.010 μ F, A055 - 0.001 μ F,
A057 - 0.001 μ F, A061 - 0.001 μ F.

Potentiometer and delay flop settings are as follows.

P001: .0010	P043: .0500	DF00: 500 microseconds
P003: .5000	P044: .5000	DF03: 75 microseconds
P007: .5000	P400: .1000	DF04: 50 microseconds
P012: .5000	P401: .8700	DF05: 25 microseconds
P017: .7350	P402: .1000	DF06: 15 microseconds
P021: .5000	P404: .5000	DF07: 50 microseconds
P022: .5000	P405: .5000	DF10: 25 microseconds
P023: .5000	P406: .5000	DF12: 500 microseconds
P024: .5000	P411: .8600	DF13: 500 microseconds
P031: .5000	P415: .2000	
P032: .2000	P424: .3550	
P037: .5000		

V. SYSTEM TESTING

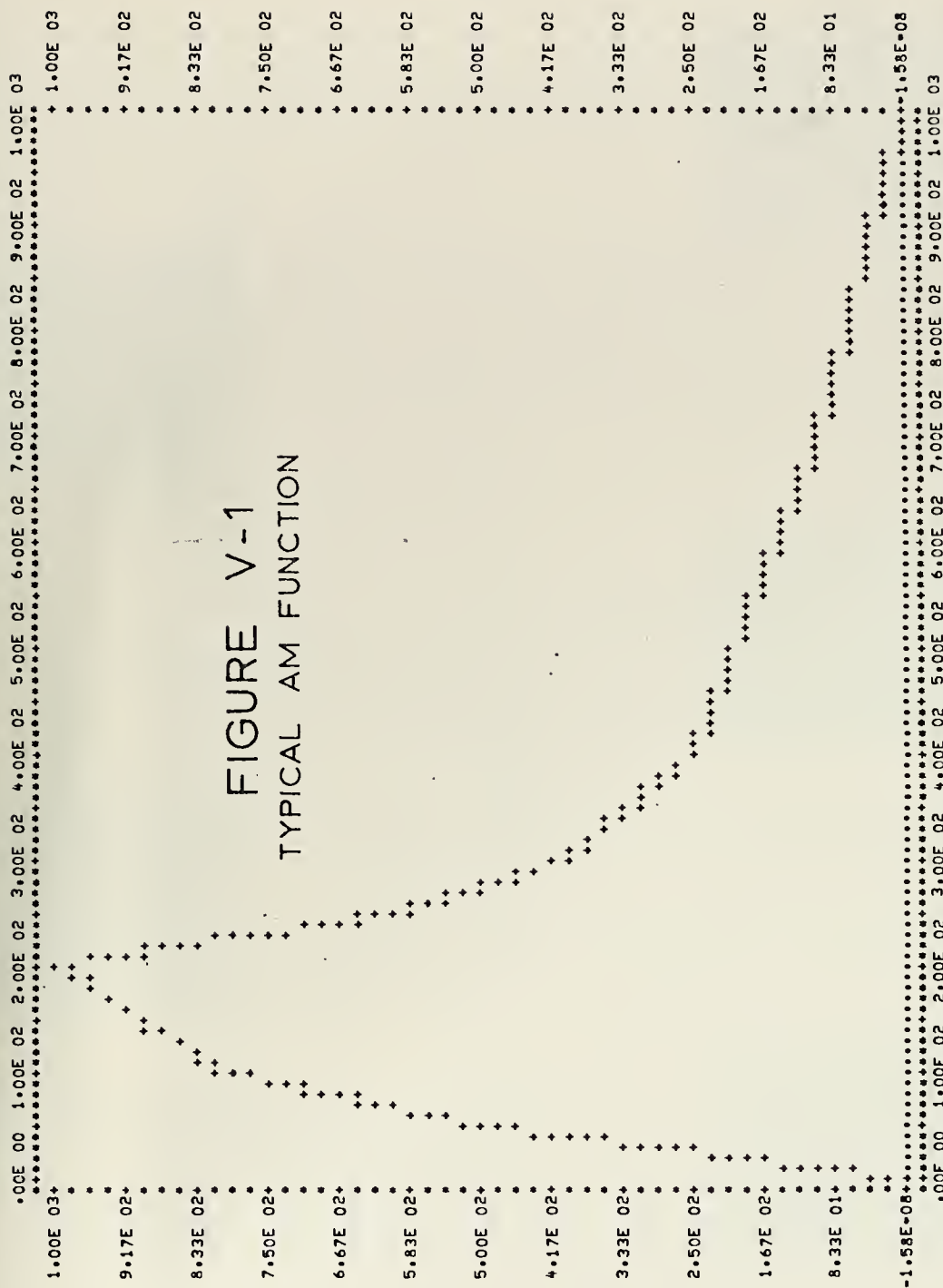
A. SPECIFIC EXPERIMENTS

Two simple experiments were attempted in order to test the operation of the system.

1. Music Synthesis

In the first experiment, some simple sequences were synthesized to create musical melodies. The purpose of this experiment was to test the technical operation of the system rather than to evaluate its practical utility. The A.M. function was used to simulate the attack and decay characteristics of a musical instrument and the periodic wave function was used to set the harmonic content of the steady state signals. The frequency elements contained the pitch information and were defined to form an equally tempered scale. Samples of typical functions used are shown in Figures V-1 and V-2. Photographs of typical waveforms at various places in the analog computer circuitry are shown in Figure V-3. The output of the analog computer was connected to an amplifier and a loudspeaker.

The system functioned properly and the programming of simple melodies proved to be fairly easy. While specific objectives other than to verify the technical operation of the system were not rigorously pursued, experimenting with the system in the area of music synthesis did tend to show some advantages and disadvantages of the system. On the



X(1)= 0 X(2)= 50 X(3)= 100 X(4)= 200 X(5)= 250 X(6)= 300 X(7)= 400 X(8)= 700 X(9)= 1000
Y(1)= 0 Y(2)= 500 Y(3)= 800 Y(4)= 1000 Y(5)= 600 Y(6)= 400 Y(7)= 250 Y(8)= 100 Y(9)= 0

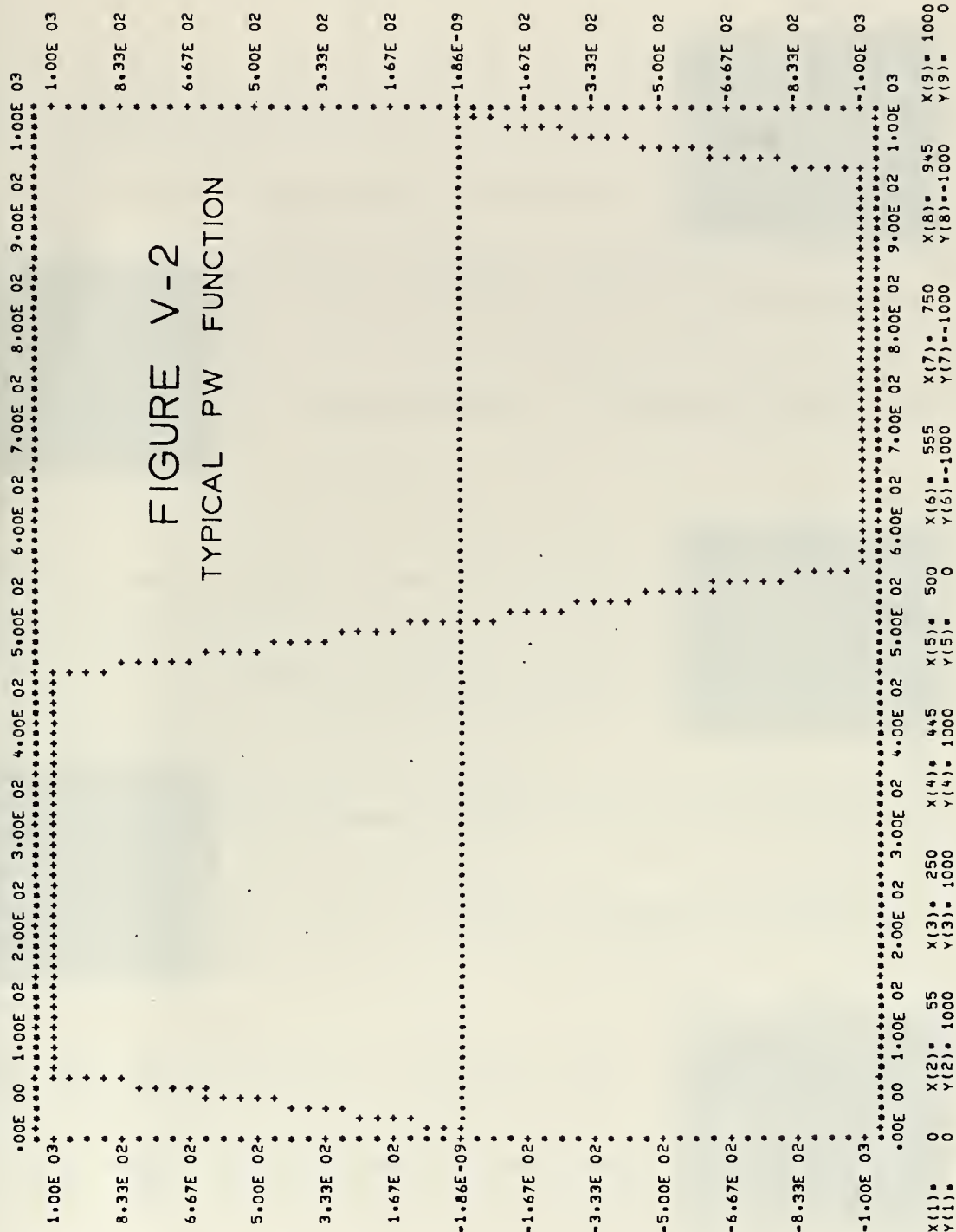
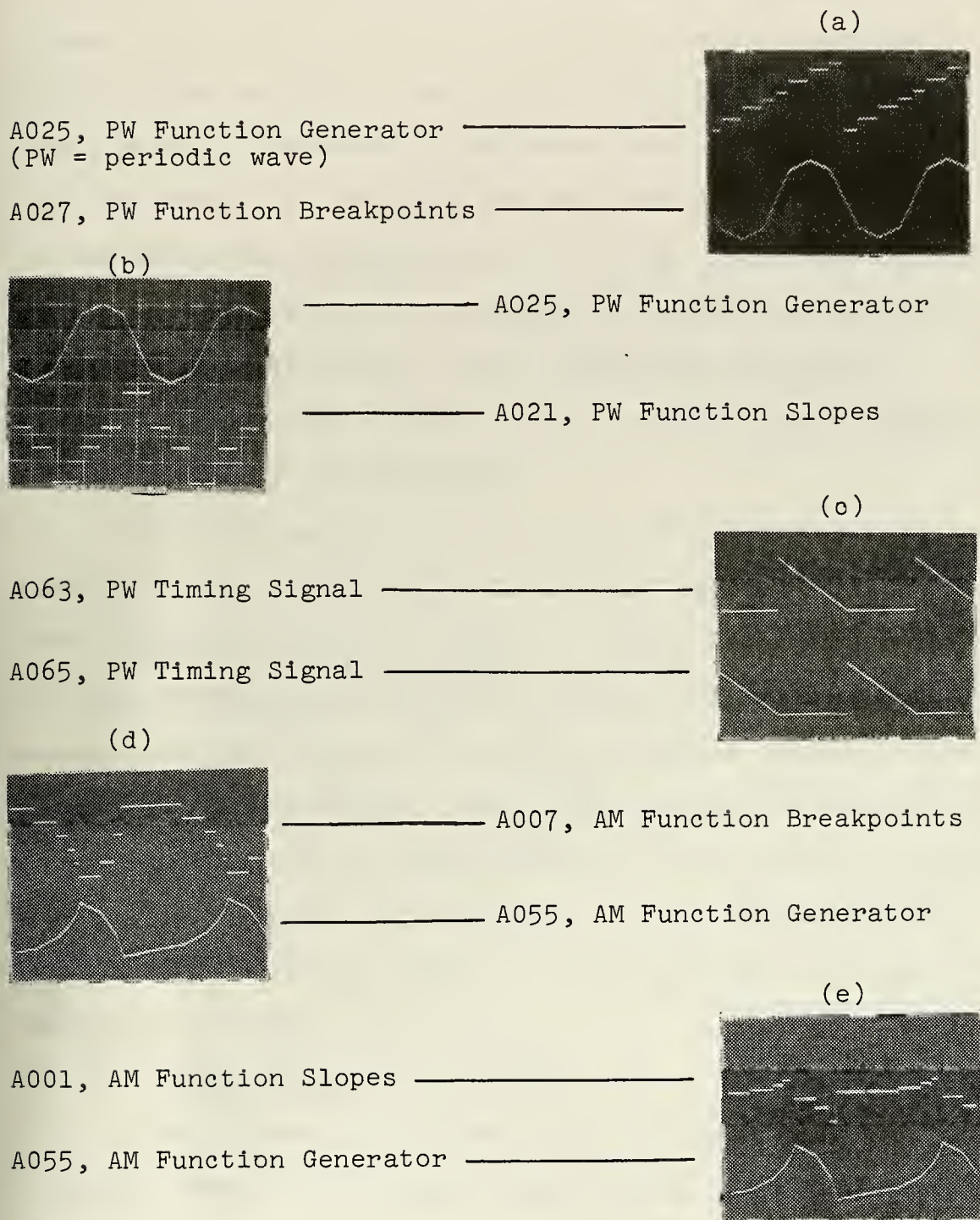


FIGURE V-3 WAVEFORMS FROM ANALOG CIRCUITS



NOTES: (1) Periodic Wave Function frequency = 200 Hz;
 AM Function Frequency = 10 Hz.
 (2) Time axis is from right to left.

positive side, the utility of the interactive and real-time capabilities of the system became evident. Being able to change one or more of the parameters describing a sound (such as the periodic wave or A.M. functions) and then being able to immediately synthesize the modified sound was very useful in relating the significance of the change to the difference in the perceived sound. However, if the system were to be used exclusively for music synthesis, a specialized input device (such as the keyboard used in Ashton's interactive organ computer communication network [2]) would be more appropriate.

On the negative side, working with fixed periodic functions was a disadvantage in attempting to synthesize known musical instrument sounds. In many natural musical instrument sounds, an important factor in providing the naturalness and distinctive quality of these sounds is the element of inharmonicity. This inharmonicity is reflected in changing time domain waveforms from pitch period to pitch period. In addition, the scarcity of data adequately describing the individual harmonics (as a function of time) further complicates the problem.

2. Single Equivalent Formants

In the second experiment, an attempt was made to synthesize voiced vowel-like sounds using single formant structures believed to be approximately perceptually equivalent to the multiformant structures of naturally generated vowel sounds. The synthesis was attempted using the limited

information on single equivalent formants contained in Ref. 15 and Ref. 27. In Ref. 15, Focht has shown that a single equivalent formant combined with voicing and amplitude information can be used to represent all speech sounds. Reference 27 shows the relationship between the single equivalent formants and the multiple formants for various phonemes and was used as the basis for the experiment.

The waveform shown in Figure V-4 (whose frequency spectrum is shown in Figure V-5) was amplitude modulated by the sinusoid approximation shown in Figure II-1. The objective was to translate the spectrum of the first waveform through the frequency domain as exemplified by Figures V-6 and V-7. The frequency of the modulating waveform was set at about 150 hz to correspond to a typical human voice pitch. The resulting sounds were recorded and later observed.

The evaluation of the sounds was mostly subjective. The resulting sounds were not judged to be good approximations of natural voiced vowel sounds, but were at least thought to be recognizable approximations. For example the "U" phoneme sounded more like that phoneme than others. It should be pointed out that the sounds were observed as isolated phonemes rather than being placed in the context of words. Since the perception of vowel sounds is partially dependent on cues provided by adjoining phonemes, it is entirely possible that a word context may have provided a more favorable evaluation.

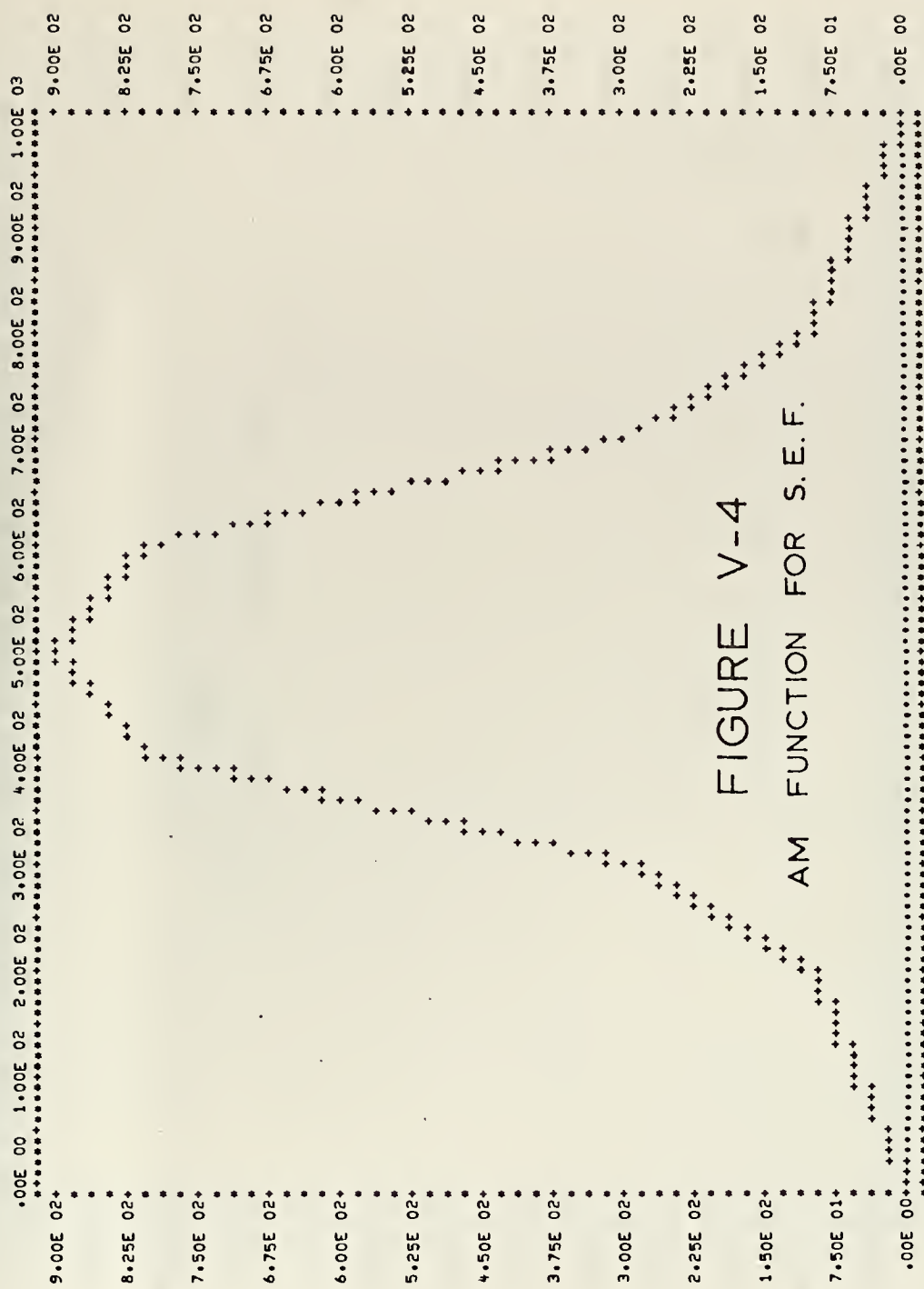


FIGURE V-4
AM FUNCTION FOR S.E.F.

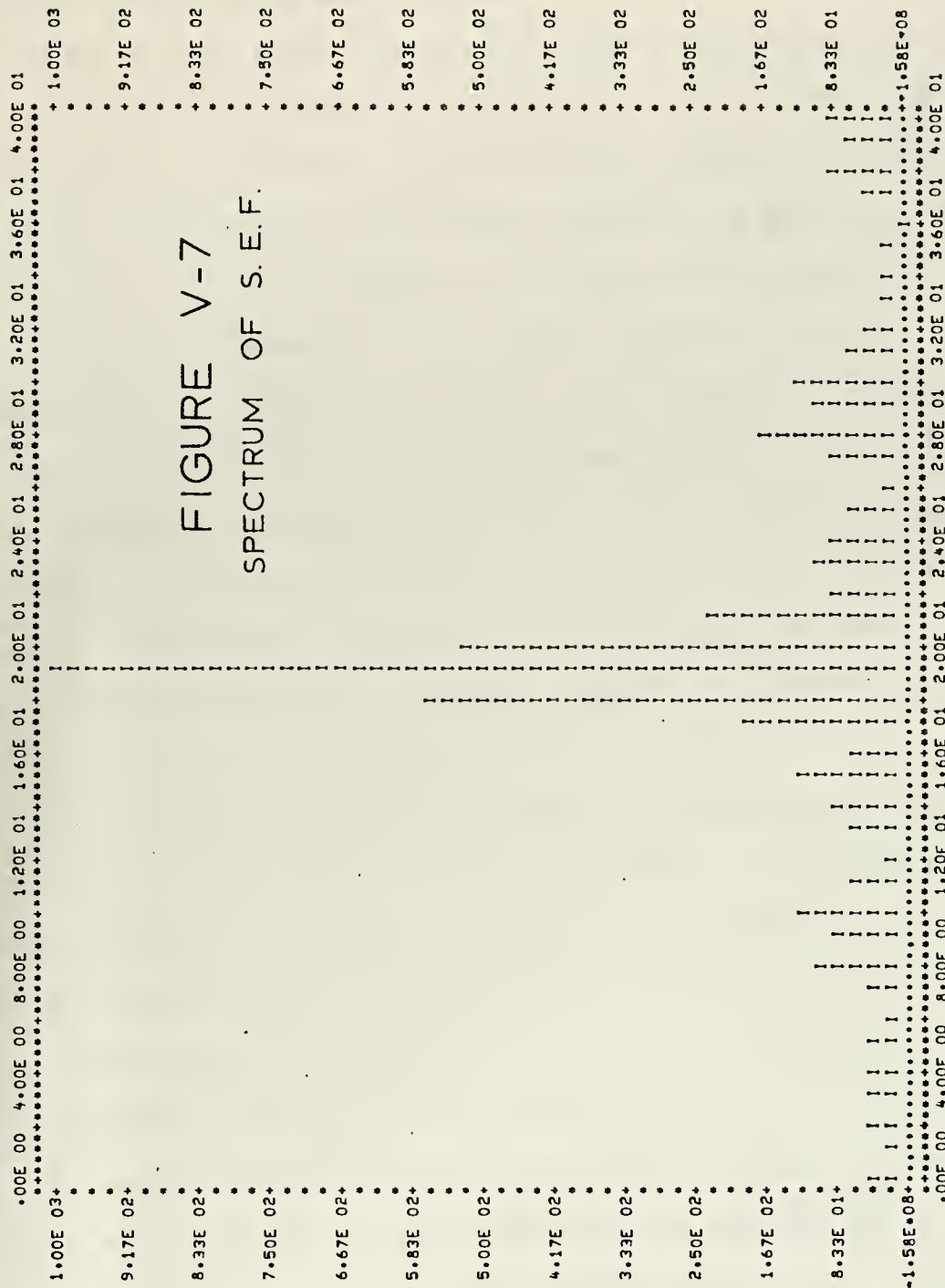
X(1)= 0 X(2)= 200 X(3)= 300 X(4)= 400 X(5)= 500 X(6)= 600 X(7)= 700 X(8)= 800 X(9)= 1000
Y(1)= 0 Y(2)= 100 Y(3)= 300 Y(4)= 800 Y(5)= 900 Y(6)= 800 Y(7)= 300 Y(8)= 100 Y(9)= 0



FIGURE V-5
 SPECTRUM OF FIGURE V-4



FIGURE V-6
 SPECTRUM OF S.E.F.



B. GENERAL OBSERVATIONS ON TESTING

In Section I-C, three types of function generators used in the system were described. To review, in the first type, two D/A channels were used. The slope and breakpoint of one segment were transmitted over these channels and then stored by analog devices. Once stored, the same channels were used to transmit the data for the next segment. In the second type, four channels were used. Two channels were used for one segment and two channels were used for the next segment. Analog switching devices were used to switch from one segment to the next. In the third type, eight channels were used. All segment slopes were transmitted and then stored by analog devices. The same channels were then used to transmit segment breakpoints.

The two specific experiments previously described demonstrated some of the characteristics of these three types of function generator circuits. The first type proved to be useful for functions whose frequencies were below about 75 Hz. The primary reason for this was the time required for the storage devices to track new values. Also, in this type, the segment breakpoint was detected by comparing the amplitude of the segment being generated to the breakpoint amplitude. Because of this, errors introduced by the digital quantization of analog values and the nonlinearities of the analog components resulted in error in the detection of breakpoints. During one period of the function, the

errors for individual segments were cumulative, resulting in a cumulative function frequency error. In general, the amount of error was proportional to the frequency of the function.

In the second type of function generator, where four channels were used, function frequencies up to about 200 Hz were possible because the circuit did not have to track segment values but only had to switch from one set of values to another. Breakpoint detection was also by an amplitude comparison method, which resulted in frequency distortion.

In the third type of function generator, function frequencies up to about 1900 Hz were possible. This type of function generator used a comparison between segment time and breakpoint time to detect breakpoints. As a result, the distortion of function frequency was minimized but at the expense of distortion of segment amplitudes. The primary disadvantage of the third type of function generator was the time required to switch from one function to another. While the time was less than a millisecond, it was still very significant from a perceptual standpoint, since the "dead time" in between functions is noticable aurally.

All three types of function generators require that the integrators that generate the segments be reset between functions. In the first two types, however, this would not be a problem if the switching between functions were performed where a segment value coincided with the initial condition of the next function. In the third type, however,

a delay is still necessary since all function slopes must be transmitted before the breakpoints. If enough D/A channels were available to transmit function slopes and breakpoints over different channels, this problem could be avoided.

At this point, some distinctions should be made in the causes of the limitations of the three types of function generators used. In the operation of the system, all three function generators operate simultaneously and the control of the generators is dependent on the digital as well as the analog computer programming. Thus, limitations result partially from the efficiency of the digital program. Other limitations are the result of the electrical capabilities of the analog components, such as tracking and recovery times. The development of specialized hardware or more efficient digital coding could improve overall efficiency without compromising the perceived quality.

VI. CONCLUSIONS AND RECOMMENDATIONS

The system displayed considerable utility as an electroacoustical tool for the investigation of psychoacoustical phenomena and as an instructive tool. It also showed itself to be useful in the investigation of time domain synthesis using first order segment functions. Its interactive capability, graphical I/O, and real-time audio output were some of its more significant attributes.

The interactive nature of the system control functions, used to direct the hybrid computer system to generate real-time audio output, was demonstrated to be effective in experiments which generated musical and voice-like signal sequences.

The system was, however, not without important limitations. The functions synthesized by the system were limited to eight segments each. The system's utility would be significantly improved if functions with a variable number of segments were permitted. The overall capability of the system was also restrained by the limited number of analog computer components and D/A converter channels available.

An inherent disadvantage of the system's synthesis technique involves the control of the output. Conventional digital synthesis (using equally spaced first order segments) is essentially stable. Synthesis using higher order and/or unequally spaced segments imposes the requirement that the

output be monitored by the system in order to detect segment breakpoints.

Because of the difficulties arising from the detection of breakpoints and the utilization of a limited number of general purpose analog computer components, it is concluded that practical application of the technique would be significantly improved with the development of specialized hardware.

Utilization of a first order segment time domain synthesis technique is also dependent on being able to derive relatively simple approximations of more complex waveforms. Perceptually equivalent audio signals can be represented by a number of different time domain waveforms. For example, it has been shown that the phase relationship of the frequency components of speech sounds is not important, but one particular relationship may result in the most simple time domain waveform. Developing mathematical techniques to minimize certain time domain parameters, such as the second and higher order derivatives, would be useful in developing simple time domain models. However, the final evaluation would still have to be made from a perceptual rather than a mathematical viewpoint. Another analytical problem in deriving the time domain functions involves separating the envelope, or amplitude modulation function, from the composite signal. Since the spectrum of the envelope usually involves frequencies much less than those of

the "carrier," a mathematical technique such as cepstral analysis could be applied.

A final area believed worthy of investigation is that of combining the time domain synthesis technique with a frequency domain technique. The parameters that are better described in each domain could then be controlled by the corresponding technique. For example, in conventional speech synthesis, formant and pitch trajectories (as functions of time) could easily be controlled by first order segment functions.

APPENDIX A
SYSTEM START UP

Prior to putting the system into operation, the following steps must be taken. Potentiometers and delay flops should be set as indicated in Section IV-0. The analog computer should then be placed in the following state: POTSET, TIME SCALE, DIGITAL CONTROL.

If the system is loaded from a FORTRAN source tape, it should be mounted on tape unit two (MT2A); if it is loaded from a core image tape, it should be mounted on tape unit three (MT3A). The tape containing user definitions to be initially loaded into the system should be mounted on tape unit one (MT1A). Sense switch number two should be on.

Using the core image tape, the following control cards are required.

```
$PATCH  
$>>DATA  
077000 31677774  
077001 04037730  
077002 10477776  
077003 17577774  
077004 04037711  
077005 00135677  
$END  
ΔJOB  
ΔAGT  
ΔASSIGN X1=MT3A  
ΔRERUN
```

When the system's programs are compiled using a FORTRAN source tape, the normal XDS 9300 system control cards should be preceded by the following.


```

$PATCH
$>>DATA
004407 01000000
004430 00104273
$9GETBUFF
000200R 00177000
077000 31677774
077001 04037730
077002 10477776
077003 17577774
077004 04037711
077005 00100203 $9GETBUFF
$END
ΔAGT
ΔASSIGN SI=MT2A
ΔASSIGN 10=DF1A
ΔASSIGN 11=DF1A
ΔASSIGN 12=DF1A
ΔASSIGN 13=DF1A
ΔASSIGN 14=DF1A
ΔASSIGN 15=DF1A
ΔASSIGN 18=DF1A
ΔASSIGN 51=DF1A
ΔASSIGN 52=DF1A
.
.
.
.
ΔASSIGN 75=DF1A

```

When the program begins execution, it will type the following message on the XDS 9300 teletypewriter:

IF STORAGE TAPE NEW, TYPE NEW; IF OLD, TYPE OLD.

An "old" tape means one that has user definitions to be initially loaded into the system. After the user responds, the system will remind the user (using the XDS teletypewriter) to make sure the analog computer has been properly prepared and will also ask the user to specify which of the AGT's is to be used. The program GATED should have been previously loaded and executed in the AGT to be used.

Following this, system operation is controlled at the AGT.

[illegible]

U


```

* * 4HF.M.,4H FUN,4HCTI0,4HN ,
* * 4HAMPL,4HITUD,4HE ,4H ,
* * 4HFREQ,4HUENC,4HY ,4H ,
* * 4HAMP.,4HFUNC,4HT. S,4HCALE,
* * 4HFREQ,4H.FUN,4HCT.S,4HCALE,
* * 4HDURA,4HTI0N,4H ,4H ,
* * 4HSIGN,4HAL S,4HEQUE,4HNCE ,
* * 4HSING,4HLE F,4HUNCT,4HI0N ,
* * 4HPW X,4H AM ,4H ,4H ,
* * 8*4H ,
* * 4HTYPE,4H RAT,4HI0 X,4H 100,
* * 4HELEM,4HENT,,4H DEF,4HINE ,
* * 4HDELE,4HTE S,4HIGAN,4HL ,
* * 4HINSE,4HRT S,4HIGNA,4HL ,
* * 4HST0R,4HE SE,4HQUEN,4HCE ,
* * 4HFIND,4H SPE,4HCTRU,4HM ,
* * 4HEXEC,4HUTE ,4H ,4H ,
* * 4HDEFI,4HNE S,4HIGNA,4HL ,
* * 4HRECA,4HLL S,4HEQUE,4HNCE ,
* * 4HPRIN,4HT ,2*4H ,
* * 4HINPU,4HT A ,4HCHOI,4HCE ,
* * 4HINPU,4HT A ,4HNUMB,4HER ,
* * 4HINPU,4HT A ,4HVALU,4HE ,
* * 4H-999,4H9+AM,4HP+9,4H999 ,
* * 4H*INP,4HUT E,4HREQU,4HENCY,
* * 4H50 +,4HFREQ,4H + ,4H999 ,
* * 4HDISP,4HLAY ,4HSIGN,4HAL ,
* * 4HAMP,,4H PRI,4HNT ,4H ,
* * 4HFREQ,4H, PR,4HINT ,4H ,
* * 4HSIG ,4HSEQN,4H, PR,4HINT ,
* * 4HP.W.,4H, PR,4HINT ,4H ,
* * 4HA.M.,4H, PR,4HINT ,4H ,
* * 4HF.M.,4H, PR,4HINT ,4H ,
* * 4HPW X,4H AM,,4H PRI,4HNT ,
* * 12*4H /

```


C C

```
DATA IB8X/00000000B,60306030B,60301751B,17501751B,17506031B,
* 60306031B,60300000B,17500001B/
DATA IGRID/00000000B,17506340B,60306341B,60306650B,17506651B,
* 17507160B,60307161B,60307470B,17507471B,17500310B,60300311B,
* 60300620B,17500621B,17501130B,60301131B,60301440B,17501441B,
* 11301750B,11306031B,03106030B,03101751B,74701750B,74706031B,
* 66506030B,66501751B,63401750B,63406031B,71606030B,71601751B,
* 00001750B,00006031B,06206030B,06201751B,14401750B,14406031B/
DATA ILIN/36,37,38,39,40/
DATA BPFAC/3*9.99/SLFAC/3*1000.0/SLMAX/3*9999.0/
DATA SLMIN/0.0,2*25.0/
```

C

```
DATA ICHAN/257,258,259,260,
* 261,262,263,264,
* 266,265,275,268,
* 276,277,274,267,
* 273,272,278,279/
DATA LOGICT/00000000B,60306030B,60307471B,60300000B,60301751B,
* 17500000B,60300001B,17506650B,60306651B/
DATA KDISP/10*(63401357B,70141751B,74701357B,00000001B,
* 03106423B,07646031B,14406423B,17500001B),
* 10*(63401357B,70141751B,74701357B,00000001B,
* 03106423B,07646031B,14406423B,17500567B),
* 10*(63401357B,70141751B,74701357B,00000001B,
* 03106423B,07646031B,14406423B,17500001B)/
```

C C

C C C C

```
NPT=512
CALL FTINIT(NEWOLD)
```



```

CALL TAPEDRUM(NEWOLD)
C
C 1195 CONTINUE
C
CALL SHSIG(0,1,-1,1,9)
CALL SHSIG(0,-1,0,39,40)
L23=23
C.....DISPLAY OPERATION
C
CALL SHSIG(0,1,0,15,L23)
C
C.....REQUEST CHOICE
C
CALL SHSIG(0,-1,0,24,24)
1199 CALL OUTIN(1,NND,10)
C
1801 FORMAT(1R4)
C
DECODE(4,1801,KOUTIN)IWORKA(1) ..
C
DO 1200 I=1,20
IF(IWORKA(1).EQ.KCMD(I)) GO TO 1201
1200 CONTINUE
GO TO 1195
C
1201 CONTINUE
C
C.....BLANK OPERATION CHOICES
C
CALL SHSIG(0,1,-1,15,L23)
GO TO (7001,.....,7009,9009,1251,.....,1260) I
C
1251 CALL USER

```



```

      GO TO 1195
1252 CALL CLEAR
      GO TO 1195
1253 CALL TAPEDRUM(3)
      GO TO 1195
1256 CONTINUE
1254 CALL PRNTIT(K,IWORKA(1))
      GO TO 1195
1255 CALL BITREC(KZ,IWORKA,KZ,KZ)
      GO TO 1195
1257 CALL TAPEDRUM(1)
      GO TO 1195
1258 CONTINUE
1259 CONTINUE
1260 CONTINUE
      GO TO 1195

C.....DEFINE SECTION
C
7001 CONTINUE
C
C.....DISPLAY CHOICES
C
      CALL SHSIG(0,1,0,1,5)
1203 CALL GUTIN(1,NND,10)
      DECODE(4,1801,KGUTIN)IWORKA(2)
C
      K=0
C
      DO 1210 I=1,5
      K=K+1
      IF(IWORKA(2).EQ.NOTES(1,I)) GO TO 1211
1210 CONTINUE
C
C.....CHOICE NOT RECOGNIZED, TRY AGAIN

```



```

C      GO TO 1203
C
C      1211 CONTINUE
C
C.....BLANK CHOICES NOT DESIRED
C
C      CALL SHSIG(0,1,-1,1,5)
C      CALL SHSIG(0,-1,-1,1,2)
C      IF(K.EQ.5) GO TO 1230
C
C.....REQUEST INPUT OF NUMBER
C
C      2212 CONTINUE
C      CALL SHSIG(0,1,0,K,K)
C      CALL SHSIG(0,-1,0,25,25)
C
C      1212 CALL OUTIN(1,NND,10)
C      IF(NND.EQ.5) GO TO 1195
C      IF(NND.NE.1) GO TO 1211
C      BUTTEN(2)=0
C      DECODE(8,1805,KOUTIN)NR,BUTTEN(2)
C      IF(NR.GT.9.6R.NR.LT.0) GO TO 1212
C
C.....BLANK REQUEST AND CHOICE
C
C      CALL SHSIG(0,-1,-1,25,25)
C      CALL SHSIG(0,1,-1,K,K)
C      IF(K.GT.3) GO TO 1220
C
C.....DEFINE WAVE, AMP FUNCT, OR FREQ FUNCT
C
C      IF(BUTTEN(2).NE.4H000N)CALL DEFINE(NR,K,0)
C      IF(BUTTEN(2).EQ.4H000N)CALL DEFINE(NR,K,0,K)
C      GO TO 2212

```



```

1802 FORMAT(114)
1805 FORMAT(114,1R1)
C
1220 CONTINUE
C
C.....DEFINE AMPLITUDE
C
      CALL SHSIG(0,-1,0,26,27)
1221 CALL OUTIN(1,NND,10)
      IF(NND.EQ.5) GO TO 1195
      IF(NND.NE.1) GO TO 1221
      DECODE(4,1802,KOUTIN)IW8RKA(3)
      IF(IABS(IW8RKA(3)).GT.9999) GO TO 1221
      R=IW8RKA(3)*1.6384
      T16=INUM(R,1)
      AMP(NR)=T16*2**9+ICHAN(15)
      GO TO 1211
C
C.....DEFINE FREQUENCIES
C
1230 CONTINUE
      CALL PRNTIT(K,K)
      GO TO 1195
C
C.....DISPLAY SIGNAL
C
8009 CONTINUE
      IF(NND.EQ.5) GO TO 1195
      IF(NND.NE.1) GO TO 1311
      DECODE(4,1802,KOUTIN)NR
      IF(NR.LT.1.0R.NR.GT.1BT0P-1) GO TO 1311
      CALL SHSIG(NR,+1,1,1,18)
      GO TO 7009

```



```

7002 CONTINUE
7003 CONTINUE
7004 CONTINUE
7008 CONTINUE
1311 CONTINUE
      CALL SHSIG(0,1,0,1+14,1+14)
7009 CONTINUE
      CALL SHSIG(0,-1,0,25,25)
      CALL OUTIN(1,NND,10)
      IF(NND.EQ.5) GO TO 1195
      IF(NND.NE.1) GO TO 7009
      DECODE(4,1802,KOUTIN)NR
      GO TO (1195,8002,8003,8004,1195,1195,8008,8009)I
C
C
C
C.....RETRIEVE SIGNAL SEQUENCE
C
      8008 ENCODE(48,1780,TXT(1,6))
           ENCODE(48,1781,TXT(1,7))
C
      1780 FORMAT($ IF DATA NOT DESIRED, PRESS CARRIAGE RETURN $)
      1781 FORMAT($ IF DESIRED,TYPE ANYTHING AND CARRIAGE RETURN $)
C
      DO 1315 I=1,6
      1315 IWORKA(I)=1
C
      K=0
C
      KKK=LLL=9
C
      DO 1325 J=1,2
C
      DO 1320 I=KKK,LLL
      K=K+1

```



```

CALL SHSIG(0,1,0,1,1)
CALL OUTIN(1,NND,10,6,7)
IF(NND.EQ.5) GO TO 1321
IF(NND.EQ.3) GO TO 1320
IWORKA(K)=0
1320 CONTINUE
C
    KKK=1
    LLL=5
1325 CONTINUE
1321 CONTINUE
C
    CALL STRET(+1,NR,IWORKA)
1322 CONTINUE
    CALL OUTIN(-1,NND,10,6,7,10)
    GO TO 1195
C
C.....STORE SIGNAL SEQUENCE
C
8004 CALL STRET(-1,NR,KOUTIN)
    GO TO 1195
C
C.....DELETE SIGNAL
C
8002 CONTINUE
    IF(NR.GT.IBTOP-1,OR.NR.LT.1) GO TO 1195
    IF(NR.EQ.IBTOP-1) GO TO 172
C
    DO 171 I=NR-1,1,-1
    CALL BITREC(I,IBIT,ISUM)
    IF(ISUM.EQ.I8) GO TO 172
171 CONTINUE
C

```



```

CALL BITREC(NR+1,IBIT,ISUM)
IF(ISUM.EQ.18) GO TO 172
GO TO 1195
172 CONTINUE
C
L8C=ISIG(NR)/4096
C
CALL BITREC(NR,IBIT,ISUM)
IF(NR.EQ.IBT0P-1) GO TO 175
C
DO 170 I=NR+1,IBT0P-1
KL8C=ISIG(I)/4096
KBIT=ISIG(I)-KL8C*4096
ISIG(I-1)=(KL8C+ISUM)*4096+KBIT
170 CONTINUE
C
175 CONTINUE
C
DO 180 I=L8C-ISUM,ITT0P-1,-1
ISIG(I+ISUM)=ISIG(I)
180 CONTINUE
IBT0P=IBT0P-1
ITT0P=ITT0P+ISUM
C
GO TO 1311
C
C.....INSERT SIGNAL
C
8003 CONTINUE
IF(NR.LT.1.0R.NR.GE.IBT0P-1) GO TO 7009
IF(ITT0P-IBT0P.LT.18.0R.IBT0P.LE.1) GO TO 1195
LOGICS(32)=0
CALL SIGDEF(IBT0P,NR,1)

```



```

IF(LOGICS(32).EQ.0) GO TO 1311
CALL BITREC(1BT0P,1BIT,ISUM)
T16=ISIG(1BT0P)-(ISIG(1BT0P)/4096)*4096
T26=(ISIG(NR+1)/4096)*4096
KS2=T16+T26
C
ML0C=ISIG(NR+1)/4096
C
DO 1925 I=1TT0P+1,ML0C
  ISIG(I-ISUM)=ISIG(I)
1925 CONTINUE
C
DO 1920 I=1BT0P,NR+2,-1
  T16=ISIG(I-1)
  ISIG(I)=T16-4096*ISUM
1920 CONTINUE
C
K=1BT0P+18
C
DO 1930 I=ML0C,ML0C-ISUM+1,-1
  ISIG(I)=ISIG(K)
  K=K-1
1930 CONTINUE
C
1BT0P=1BT0P+1
1TT0P=1TT0P-ISUM
ISIG(NR+1)=KS2
C
GO TO 1311
C
C.....DEFINE SIGNAL
C
7007 CONTINUE
C

```



```

      CALL SHSIG(0,-1,-1,24,24)
      CALL OUTIN(-1,NND,10,10)
8007 CONTINUE
C
      CALL SIGDEF(1BTOP,NEXSIG,+2)
      GO TO 1195
C
C
C.....EXECUTE
C
7006 CONTINUE
      CALL SIGOUT
      GO TO 1322
C
C
C.....FIND FREQUENCY SPECTRUM
C
7005 CONTINUE
C
C
C.....REQUEST CHOICE, SINGLE FUNCT, OR PW X AM
C
      CALL SHSIG(0,-1,0,24,24)
      CALL SHSIG(0,1,0,10,11)
C
6700 CALL OUTIN(1,NND,10)
      IF(NND.EQ.5) GO TO 1195
      DECDE(4,1801,KOUTIN)IWORKA(1)
C
      DO 6701 I=10,11
      K=I
6701 IF(IWORKA(1).EQ.NOTES(1,I)) GO TO 6702
      GO TO 6700
C

```



```

6702 IF(K.EQ.11) GO TO 6770
C
C
C.....SINGLE FUNCTION: PERIODIC WAVE OR A.M. FUNCTION
C
      KPART(1)=KPART(2)=1000
      IWAY=1
      CALL SHSIG(0,1,0,1,2)
6703 CALL OUTIN(1,NND,10)
      DEC9DE(4,1801,KOUTIN)IWORKA(1)
C
      DO 6704 I=1,2
      MODE=1
6704 IF(IWORKA(1).EQ.N0TES(1,1)) GO TO 6705
C
      GO TO 6703
C
C.....DETERMINE NUMBER
C
C
6705 CONTINUE
      CALL SHSIG(0,-1,0,25,25)
      CALL SHSIG(0,1,-1,25,25)
      CALL SHSIG(0,1,0,MODE,MODE)
C
6706 CALL OUTIN(1,NND,10)
      IRET=2
      IF(KOUTIN(1).EQ.4HPART) GO TO 6790
      IF(NND.NE.1) GO TO 6706
      DEC9DE(4,1802,KOUTIN)NR
      IF(NR.LT.0.OR.NR.GT.9) GO TO 6706
      GO TO (6709,6772,6773)IWAY
6709 CONTINUE
      RAT=1.0
      NR1=NR

```



```

C 6773 CONTINUE
CALL SHSIG(0,1,-1,40,40)
NR2=NR
MODE=1
IWAY=2

C.....DETERMINE RATIO
C
CALL SHSIG(0,-1,0,14,14)
6778 CONTINUE
KPART(1)=KPART(2)=1000
6780 CONTINUE
IRET=1
ENCODE(48,6774, TXT(1,6))
6774 FORMAT($ 1 + RATIO + 20, TYPE PART FOR PART OPTION $)
6775 CALL OUTIN(1,NND,10,6)
IF(KOUTIN(1).EQ.4HPART)GO TO 6790
IF(NND.NE.1) GO TO 6775
DECODE(4,1802,KOUTIN)KRAT
1803 FORMAT(118)
IF(NND.NE.1) GO TO 6775
IF(KRAT.GT.2000.0R.KRAT.LT.100) GO TO 6775
RAT=KRAT/100.0
GO TO 6710
6790 CONTINUE
D8 6795 L=1,2
ENCODE(48,1804, TXT(1,6))L
1804 FORMAT($ INPUT PART $,111,$, 10<PART<1000 $)
6791 CALL OUTIN(1,NND,10,6)
CALL OUTIN(-1,NND,10,6)
KPART(L)=1000
IF(NND.EQ.5) GO TO (6778,6706,9021)IRET
IF(NND.NE.1) GO TO 6791
DECODE(4,1802,KOUTIN)
(L)

```



```

        IF(KPART(L).LT.10.0R.KPART(L).GT.1000) GO TO 6791
        6795 CONTINUE
C
        GO TO (6780,6706,9021) IRET
C
C.....PRINT SECTION
C
C
        9009 CONTINUE
        CALL SHSIG(0,-1,0,24,24)
        CALL SHSIG(0,1,0,31,37)
        9010 CALL BUTIN(1,NND,10)
        CALL SHSIG(0,1,-1,30,37)
        IF(NND.EQ.5) GO TO 1195
C
        K=1
        DECODE(4,1801,K0UTIN)IWRKA(1)
C
        IF(IWRKA(1).EQ.4HC0MM) GO TO 9013
        DO 9011 I=31,37
        K=K+1
        IF(IWRKA(1).EQ.N0TES(1,I))GO TO 9012
        9011 CONTINUE
C
        GO TO 9009
C
        9012 CONTINUE
        CALL BUTIN(-1,NND,10,10)
        IF(K.GT.4) GO TO 9020
        CALL SHSIG(0,-1,-1,24,24)
        CALL PRNTIT(K)
C
        GO TO 9009
C
C

```



```

9013 CONTINUE
CALL BITREC(NND,IWORKA,NND,NND)
GO TO 9009
C
C
C   PLOT WAVEFORMS
9020 CONTINUE
    IWAY=1
    IF(K.EQ.8) GO TO 6770
C
    CALL SHSIG(0,-1,0,25,25)
    CALL SHSIG(0,1,-1,25,25)
    CALL SHSIG(0,1,0,K+29,K+29)
    KPART(1)=KPART(2)=1000
C
9021 CALL GUTIN(1,NND,10)
    IRET=3
    IF(KGUTIN(1).EQ.4HPART) GO TO 6790
    IF(NND.NE.1) GO TO 9021
C
C.....REQUEST NUMBER
C
    DECODE(4,1802,KGUTIN)NR
    IF(NR.LT.0.OR.NR.GT.9) GO TO 9021
C
    CALL FRESPEC(NR,NR,IWAY,1.0,K-4,0,KPART(1),KPART(2),NPT)
C
    GO TO 9009
999 STOP
END

```



```

SUBROUTINE SIGOUT
SUBROUTINE T0. CONTROL ANALOG COMPUTER SYNTHESIS OF SIGNALS.
INTEGER ASIG,BSIG,AMP,FREQ
INTEGER BUTTN,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPOX,TEMPOY
INTEGER TEMPSL,TEMPBP
COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
COMMON KFRE,KAMP,KS2,KSIG,I8,IBT0P,ITT9P
COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
* FREQ(72),KAAK,KDIV,KUP
COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
COMMON/KDS/KDISP(8,0:9,3)
COMMON/IGRAF/IBOX(8),IGRID(35),BUTTN(2),IFIG(100)
COMMON IW9KA(10),IXHOLD(15),TEMPX(10),TEMPY(10),
* TEMPOX(10),TEMPOY(10),TEMPSL(8),TEMPBP(8)
COMMON K0UTIN(15),IW9KB(15)
COMMON/INF0/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
* IDC1,IDF1

INTEGER DACOUT,DAF0UT,DCSTRT,DFSTRT
INTEGER RESET,COMPUTE
DATA RESET/4H1000/COMPUTE/4HCO00/
INTEGER SWAVE,SAM,SFM,BWAVE,BAM,BFM
DIMENSION SWAVE(8,0:9),SAM(8,0:9),SFM(8,0:9),
* BWAVE(8,0:9),BAM(8,0:9),BFM(8,0:9)
EQUIVALENCE(ASIG(1,0,1),SWAVE(1,0)),(ASIG(1,0,2),SAM(1,0)),
* (ASIG(1,0,3),SFM(1,0)),(BSIG(1,0,1),BWAVE(1,0)),
* (BSIG(1,0,2),BAM(1,0)),(BSIG(1,0,3),BFM(1,0))
DIMENSION DAC0UT(22),DAF0UT(6)
EQUIVALENCE(IFIG(1),DAC0UT(1)),(IFIG(51),DAF0UT(1))
EQUIVALENCE(IFIG(52),KDAF2),(IFIG(53),KDAF3)
EQUIVALENCE(IFIG(54),...),(IFIG(51),DFSTRT)

```


C	EQUIVALENCE(TEMPX(1),ITSTRT)
	DIMENSION IB9T(100),IT9P(100),N9TLOC(100)
C	DIMENSION KIKI(0:9)
	EQUIVALENCE(LOGICS(1),KIKI(0))
C	KIC=ICHAN(10)-ICHAN(16)
	DO 1234 I=0,9
	K=-(BAM(8,I))/(2**9))
	1234 LOGICS(I+1)=K*2**9+ICHAN(16)
C	
C	LDA \$+2
C	BRU \$+2
S	BRM 7000S
S	STA 041
S	K=0
S	EBM 033001
S	P9T K
C	
	KKSL=ICHAN(9)-ICHAN(20)
	KKBP=ICHAN(10)-ICHAN(19)
	ITSTRT=262144+IRR2
S	EBM 030015
S	P9T COMPUTE
S	SKS 030400
S	BRU \$-1
	IWUN=1
	TEMPX(10)=0
	T26=77776777B
	T36=77775777B
	DFSTRT=00200000B+IDF1
	DAF9UT(4)=0

S	E9M 033001	
S	P6T =077777037	
C	G8 T8, 8000	
C		
C	500 CONTINUE	
C		
C	KAAC=ISIG(IPLACB)	
C		
C	K=1	
C	KW=KA=KF=-1	
C	T16=KTN	
C		
C	C.....UNPACK SIGNAL PARAMETER DEFINITION AND SET UP DAC TABLE	
C		
C		
C	D8 10 I=1,8	
S	LDA KAAK	
S	STA LB	
S	SHIFT 020001	
S	STA KAAK	
S	SHIFT 024001	
S	STA KDIV	
S	LDA LB	
S	SUB KDIV	
S	STA K10	
S	SKU =0	
S	BRU 9S	
S	LDA I	
S	SKU =8	
S	BRU 9S	
S	K=K+1	
S	LDX I,2	
S	BRU \$,2	

S	BRU 2S	
S	BRU 3S	
S	BRU 33S	
S	BRU 4S	
S	BRU 5S	
S	BRU 6S	
S	BRU 6S	
C	2 CONTINUE	
C	INDX=ISIG(KUP)	
	D0 222 J=1,8	
	DACOUT(J+1)=BWAVE(J,INDX)	
	TEMPX(J+1)=SWAVE(J,INDX)	
	IF(IWUN.EQ.1) G0 T0 222	
C	LFLAG=1	
S	SKS 034001	
S	BRU \$+2	
S	BRU 5000S	
C	1222 CONTINUE	
	LFLAG=2	
S	SKS 031020	
S	BRU \$+2	
S	BRU 6000S	
C	222 CONTINUE	
C	K=9.	
C	KW=1	
	T16=T16-00000140B	
	G0 T0 8	
C	3 CONTINUE	


```

311 NEXA=INDX=ISIG(KUP)
CONTINUE
KA=1
T16=T16-00000400B
DACOUT(K)=SAM(1,INDX)
K=K+1
DACOUT(K)=BAM(1,INDX)
K=K+1
DACOUT(K)=KIKI(INDX)
K=K+1
DACOUT(K)=SAM(2,INDX)-KKS L
K=K+1
DACOUT(K)=BAM(2,INDX)-KKBP
IF(K10.EQ.0) GO TO 312
GO TO 8

C 33 CONTINUE
KF=1
NEXF=INDX=ISIG(KUP)
T16=T16-00000200B
DACOUT(K)=SFM(1,INDX)
K=K+1
DACOUT(K)=BFM(1,INDX)
GO TO 8

C 4 CONTINUE
DACOUT(K)=AMP(ISIG(KUP))
GO TO 8

C 5 CONTINUE
DACOUT(K)=FREQ(ISIG(KUP))
GO TO 8

C 6 CONTINUE
DACOUT(K)=ISIG(KUP)

```



```

C      8 KUP=KUP-1
C
C      9 CONTINUE
      IF(IWUN.EQ.1) GO TO 10
      IF(I.NE.3.OR.KA.EQ.1.OR.KMUS.NE.0) GO TO 312
      NEXA=INDX=NRA
      GO TO 311
      312 CONTINUE
C
      LFLAG=3
      SKS 034001
      BRU $+2
      BRU 5000S
C      1119 CONTINUE
      LFLAG=4
      SKS 031020
      BRU $+2
      BRU 6000S
      10 CONTINUE
C
      DCSTRT=(K-1)*32768+IDC1
      DACOUT(K+1)=0
      IF(K10.EQ.0) GO TO 9111
C.....PREPARE TIMING SIGNAL FOR SETLINES OUTPUT
C
      IRR3=ISIG(KUP)
      KUP=KUP-1
C
      9111 CONTINUE
      IF(IWUN.EQ.1) GO TO 100
C.....IF BP DETECTORS ARE TRIGGERED,DAC NEW SLOPE AND BP

```



```

C C IF(TEST(LTLTA).GE.0) GO TO 5000
C LFLAG=5
S SKS 034001
S BRU $+2
S BRU 5000S
C C IF(TEST(LTLTF).GE.0) GO TO 6000
C C
C 9112 CONTINUE
C LFLAG=6
S SKS 031020
S BRU $+2
S BRU 6000S
C 9222 CONTINUE
C C
C C.....CHECK FOR END OF SEQUENCE
C C
C C
C C IF(TEST(LTLX).GE.0) GO TO 9999
C SKS 034004
C BRU $+2
C BRU 9999S
C C
C C.....CHECK FOR END OF SIGNAL
C C
C C IF(TEST(LTLTM).GE.0) GO TO 9111
C SKS 034002
C BRU $+2
C BRU 9111S
C C
C 100 CONTINUE
C IWUN=0
C IPLACB=IPLACB+1

```



```

C C S S
EOM 033001
POT T16
IF(IPLACB.LE.IPLACT) GO TO 121
IF(KT.NE.0) GO TO 112
IF(LLL.EQ.KALT)GO TO 112
LLL=KALT
GO TO 113
112 KKK=LLL=KKK+1
IF(KKK.LE.NSEQ)GO TO 113
KKK=LLL=NS
113 CONTINUE
LDX LLL,2
LDA IBOT-1,2
STA IPLACB
LDA ITOP-1,2
STA IPLACT
LDA NOTLOC-1,2
STA KUP
121 CONTINUE
IWUN=0
STX IFLAG,0
EOM 035000
POT DCSTRT
6544 IF(IFLAG.EQ.0) GO TO 6544
IF(KW.NE.1) GO TO 119
C C 101 IF(TEST(LTLW).LT.0) GO TO 101
SKS 034010
BRU $-1
T16=T16+000000040B
C.
C.....IF NEW PERIODIC WAVE, DAC NEW SLOPES
C

```



```

S      EOM 033001
S      POT T16
C
C 117 IF( TEST(31).LT.0) GO TO 117
S      SKS 034100
S      BRU $-1
S      STX IFLAG,0
S      EOM 035000
S      POT ITSTRT
S      7654 IF( IFLAG.EQ.0) GO TO 7654
C
C      119 CONTINUE
S      EOM 033001
S      POT KTN
S      NRA=NEXA
S      NRF=NEXF
S      IF( KA.NE.1) GO TO 1919
S      EOM 033001
S      POT T26
S      KAMP=3
S      LFLAG=7
S      LFLAG=7
S      EOM 033001
S      POT KTN
S      1919 CONTINUE
C
C.....INITIALIZE AND START TIMER
C
S      EOM 033001
S      POT IRR3
S      IF( KF.EQ.1) KFRE=2
S      IF( KA.LT.0) GO TO 120
S      KTAG=1
S      120 CONTINUE

```



```

S    E0M 033001
S    P0T KTN
C
C    IF(KF•NE•1) G0 T0 500
C    LFLAG=7
S    BRU 6000S
C
C    5000 C0NTINUE
C
C    KDAF2=SAH(KAMP,NRA)
C    KDAF3=BAH(KAMP,NRA)
C    IF(KT0G•GT•0) G0 T0 5003
C    KDAF2=KDAF2-KKSL
C    KDAF3=KDAF3-KKRP
C    5003 C0NTINUE
S    STX IFLAG,0
C
C    E0M 035000
S    P0T DFSIRT
S    5006 IF(IFLAG•EQ•0) G0 T0 5006
C
S    E0M 033001
S    P0T T26
C    KAMP=KAMP+1
C    IF(KAMP•EQ•9) KAMP=LF
C    IF(KT0G•EQ•1) G0 T0 5004
C    KT0G=+1
C    G0 T0 5005
C    5004 C0NTINUE
C    KT0G=-1
C    5005 C0NTINUE
S    E0M 033001
S    P0T KTN

```


S	LDX LFLAG,2	
S	BRU \$,2	
S	BRU 1222S	
S	BRU 222S	
S	BRU 1119S	
S	BRU 10S	
S	BRU 9112S	
S	BRU 9222S	
S	BRU 500S	
C		
C		
C	6000 CONTINUE	
C		
S	KDAF2=SFEM(KFRE,NRF)	
S	KDAF3=BFEM(KFRE,NRF)	
S	STX IFLAG,0	
S	E9M 035000	
S	P8T DFSTRT	
C	6006 IF(IFLAG.EQ.0)G9 T8 6006	
C		
C		
S	SKS 031010	
S	BRU \$+2	
S	BRU \$-2	
C		
CARM BP DETECT8RS	
C		
S	E9M 033001	
S	P8T T36	
	KFRE=KFRE+1	
	IF(KFRE.EQ.9) KFRE=1	
S	E9M 033001	
S	P8T KTN	


```

C S S S S S S S S S S C C S7000 HLT
LDX IFLAG,2
BRU $,2
BRU 1222S
BRU 222S
BRU 1119S
BRU 10S
BRU 9112S
BRU 9222S
BRU 500S
C C
S7000 HLT
LDA =1
STA IFLAG
BRX *7000S,0
C.....TURN OFF ANALOG AND EXIT SUBROUTINE
C 9999 CONTINUE
C S EOM 033001
PBT =077777037
EOM 030015
PBT RESET
SKS 030400
BRU $-1
C RETURN
C C C C C
8000 CONTINUE
KT=LF=NS=K=1

```



```

NSEQ=ITE=0
CALL SHSIG(0,-1,-1,24,24)
KMUS=NS=LF=1
1819 CONTINUE
      ENCODE(48,1822,TXT(1,6))
1822 FORMAT($      INPUT OPTIONS OR C/R TO CONTINUE $)
8802 CALL OUTIN(1,NND,10,6)
      IF(NND.EQ.3) GO TO 8820
      IF(KOUTIN(1).EQ.4HENVL)LF=8
      IF(KOUTIN(1).EQ.4HLAST)NS=1000
      IF(KOUTIN(1).EQ.4HMUSI)KMUS=0
      IF(KOUTIN(1).EQ.4HALTE)KT=0
      IF(KOUTIN(1).EQ.4HRITE)ITE=1
      IF(KOUTIN(1).NE.4HC9MM)GO TO 8802
      CALL BITREC(NND,IW9RKA,NND,NND)
      GO TO 1819

8820 CONTINUE
1809 CONTINUE
      IBOT(K)=1
      ITOP(K)=IBTOP-1
      ENCODE(48,1810,TXT(1,6))
1810 FORMAT($      INPUT NUMBER OF FIRST SIGNAL OF SEQUENCE $)
      ENCODE(48,1811,TXT(1,7))
1811 FORMAT($      TYPE C WHEN COMPLETE $)
8016 CALL OUTIN(1,NND,10,6,7)
      GO TO (8017,8016,8018,8500,9999)NND
8017 CONTINUE
      ENCODE(4,1802,KOUTIN)IBOT(K)
1802 FORMAT(1I4)
      IF(IBOT(K).LT.1.0R.IBOT(K).GE.IBTOP) GO TO 8016
      IF(K.NE.1) GO TO 8018
      CALL BITREC(IBOT(K),IW9RKA,ISUM)
      IF(ISUM.NE.18) GO TO 8016
8029 CONTINUE
8018 ENCODE(48,1812,TXT(1,6))

```



```

C 1812 FORMAT($ INPUT NUMBER OF LAST SIGNAL OF SEQUENCE $)
8019 CALL OUTIN(1,NND,10,6)
    GO TO (8020,8019,8021,8019,9999)NND
8020 CONTINUE
    DECODE(4,1802,KOUTIN)ITOP(K)
    IF(ITOP(K).GE.IBTOP.0R.ITOP(K).LT.IBOT(K))GO TO 8019
8021 NSEQ=K
    N9TL8C(K)=ISIG(IBOT(K))/4096
    K=K+1
    IF(K.GT.100) GO TO 8500
    GO TO 1809
8500 LLL=KKK=1
    IF(KT.NE.0) GO TO 8022
    KALT=NSEQ
    NSEQ=NSEQ-1
8022 IF(NSEQ.LT.1) GO TO 8000
    KUP=N9TL8C(1)
    IPLACB=IB9T(1)
    IPLACT=IT9P(1)
    IF(NS.EQ.1000)NS=NSEQ
    IF(ITE.EQ.1) GO TO 9500
8023 CONTINUE
S   SKS 031001
S   BRU $-1
S   BRU 500S
9500 CONTINUE
    NND=NSEQ
    IF(KALT.GT.NSEQ)NND=KALT
    WRITE(6,9505)
9505 FORMAT($1$,6X,$SEQUENCE$,6X,$FIRST SIGNAL$,6X,$LAST SIGNAL$,/)
C
D0 9510 I=1,NND
WRITE(6,9507)I,IBOT(I),ITOP(I)
9507 FORMAT($0$,8X,1I3,12X,1I4,14X,1I4)

```


9510 CONTINUE

C

GO TO 8023
END

SUBROUTINE CLEAR

C
C
C

SUBROUTINE TO CLEAR ALL SIGNALS FROM MAIN SIGNAL SEQUENCE.

COMMON KFRE,KAMP,KS2,KSIG,K8,IBTOP,ITTOP
ITTOP=1000
IBTOP=1
RETURN
END


```

REWIND K
IF(KDKD.EQ.1)GO TO 111
C
C
C.....STORE SECTION
C
C
WRITE(K)(I0SIGA(J),J=1,1562),(I9SIGG(J),J=1,240),KEKE,IBT0P,ITT0P
RETURN
C
C
C.....RECALL SECTION
C
C
111 CONTINUE
REWIND 10,11,12,13,14,15
C
C.....STORE CURRENT DATA
C
WRITE(10)(ISIG(I),I=1,1000),NR,IBT0P,ITT0P
L=0
C
DO 10 I=11,13
L=L+1
SKS 014000
BRU $-1
WRITE(I)((ASIG(J,M,L),BSIG(J,M,L),KDISP(J,M,L),J=1,8),M=0,9)
10 CONTINUE
C
SKS 014000
BRU $-1
WRITE(14)(AMP(I),I=0,9)
SKS 014000
BRU $-1

```



```

115 WRITE(15)(FREQ(I),I=1,72)
    CONTINUE
    REWIND 10,11,12,13,14,15
    READ(K)(I0SIGA(J),J=1,1562),(I0SIGG(J),J=1,240),KEKE,IBT0P,ITT0P
C
C
    DO 200 I=1,6
    SKS 014000
    BRU $-1
    IF(I0DATA(I).EQ.1) GO TO (300,301,301,301,304,305)I
    200 CONTINUE
    RETURN
C
    300 READ(10)(ISIG(J),J=1,1000),NR,IBT0P,ITT0P
    GO TO 200
    301 READ(I+9)((ASIG(J,K,I-1),BSIG(J,K,I-1),KDISP(J,K,I-1),J=1,8),
    * K=0,9)
    GO TO 200
C
C
    304 READ(14)(AMP(J),J=0,9)
    GO TO 200
C
    305 READ(15)(FREQ(J),J=1,72)
    GO TO 200
C
C
C
C
    END

```



```

C
C FUNCTION INUM(RNM,K)
C
C FUNCTION TO PROPERLY ROUND OFF NUMBERS REPRESENTING ANALOG
C COMPUTER VOLTAGES WHEN CONVERTING BETWEEN DESIRED VOLTAGE
C VALUE AND VALUE SCALED TO CONFORM TO DIGITAL/ANALOG CONVERTER.
C K=1 WHEN ROUNDING NUMBER THAT HAS BEEN SCALED TO CONVERTER VALUE.
C K=2 WHEN ROUNDING FROM CONVERTER VALUE.
C
C   DIMENSION DD(2)
C   DATA DD/0.6,-0.6/
C   INUM=RNM+DD(K)
C   TRNM=INUM
C   IF (TRNM.LT.RNM) INUM=INUM+1
C   RETURN
C   END

```

```

C
C SUBROUTINE USER
C
C DUMMY SUBROUTINE
C
C   RETURN
C   END

```

```

C
C SUBROUTINE OUTIN(IOB,NND,LIO,/LINE(NL)/)
C
C SUBROUTINE OUTIN IS IDENTICAL TO SUBROUTINE DUTIN.
C IT IS DUPLICATED WITH A DIFFERENT NAME IN ORDER TO
C ACCOMMODATE THE PROGRAM SEGMENTATION AND OVERLAY.

```



```

C
C SUBROUTINE SIGDEF(IFIRST,NEXSIG,KDKD)
C
C SUBROUTINE USED TO DEFINE A SIGNAL
C
C IFIRST=THE NUMBER OF THE LAST SIGNAL DEFINED.
C NEXSIG=THE NUMBER OF THE SIGNAL AFTER WHICH A NEW
C SIGNAL IS TO BE INSERTED.
C KDKD: 1= ONE NOTE ONLY
C        2=MANY.
C
C INTEGER ASIG,BSIG,AMP,FREQ
C INTEGER BUTTON,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
C INTEGER TEMPSL,TEMPBP
C COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
C COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
C COMMON KFRE,KAMP,KS2,KSIG,I8,I8T8P,ITT8P
C COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
C        * FREQ(72),KAAK,KDIV,KUP
C COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
C COMMON/KDS/KDISP(8,0:9,3)
C COMMON/IGRAF/IBOX(8),IGRID(35),BUTTON(2),IFIG(100)
C COMMON IWORKA(10),IXH9LD(15),TEMPX(10),TEMPY(10),
C        * TEMPDX(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
C COMMON KOUTIN(15),IWORKB(15)
C COMMON/INFO/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
C        * IDC1,IDF1
C
C KDEF=KSH9=IFIRST
C IF(KDKD.EQ.1) GO TO 3
C KSTRT=1000
C IF(KDEF.GT.1) GO TO 3000
C CONTINUE
C KUP=KSTRT
C
C 3001 CONTINUE
C        3 CONTINUE

```



```

C      IF(KDKD.EQ.1)KUP=IBTOP+18
C
C      CALL SHSIG(2,1,0,1,18)
      IF(KDKD.EQ.1) GO TO 11
      CALL SHSIG(2,-1,0,1,18)
C
C      IF(KDEF.GT.1) CALL SHSIG(KDEF-1,-1,1,1,18)
      GO TO 1
11 CONTINUE
      CALL SHSIG(NEXSIG,-1,1,1,18)
      KSHB=NEXSIG+1
      KSTRT=KUP
C
C      1 CONTINUE
C
      ENCODE(20,1708,TXT(1,6))KSWB
      ENCODE(20,1708,TXT(1,7))KSWB-1
1708 FORMAT($ SIGNAL NUMBER $,1I4)
      CALL TEXTB(IDEV,TXT(1,6),5,3,77,1,2,IER)
      CALL TEXTB(IDEV,TXT(1,7),5,3,1,1,2,IER)
C
C      1710 FORMAT($ ***** DEFINE $,4A4,$ ***** $)
C
      IF((KUP-KDEF).LT.18) GO TO 1001
      DO 2 I=1,12
      2 IBIT(I)=0
C
C      DO 100 I=1,18
      II=I
      ENCODE(48,1710,TXT(1,6))(NBTES(J,I),J=1,4)
      15 IBIT(I)=0
      CALL BOUTIN(1,NND,10,6)
C
      GO TO (16,2116,116,1116,115)NND

```



```

115 CONTINUE
  IF(I.NE.1) GO TO 15
  IF(KDKD.EQ.1) GO TO 1001
  GO TO 1000
C.....COPY PREVIOUS SIGNAL ELEMENT
C
C
116 CONTINUE
  IF(KDEF.EQ.1) GO TO 15
  CALL TEXTI(IDEV,TEMPX,4,3*I+5,1,IER)
  TEMPX(1)=4H *
  TEMPX(2)=4H =
  CALL TEXT0(IDEV,TEMPX,4,3*I+5,81,1,2,IER)
  GO TO 100
C.....COPY REST OF PREVIOUS SIGNAL
C
C
1116 CONTINUE
C
  IF(KDEF.EQ.1) GO TO 15
  DO 1117 L=11,18
  CALL TEXTI(IDEV,TEMPX,4,3*L+5,1,IER)
  TEMPX(1)=4H *
  TEMPX(2)=4H =
1117 CALL TEXT0(IDEV,TEMPX,4,3*L+5,81,1,2,IER)
C
  GO TO 101
C
2116 CONTINUE
  IF(IW9RKB(1).NE.4H000R.0R.KDEF.EQ.1) GO TO 15
  CALL TEXTI(IDEV,TEMPY,4,3*I+5,1,IER)
  DECODE(8,1802,TEMPY(3))TEMPX(1)
  IBIT(I)=1

```



```

C      GO TO 5016
      16 CONTINUE
      IBIT(I)=1
      DEC8DE(8,1802,K9UTIN)TEMPX(1)
      1802 FORMAT(1I8)
C
      5016 CONTINUE
C
      GO TO (5021,5021,5021,5021,5025,5026,5026,5028)I
C
      5021 CONTINUE
      IF(TEMPX(1).GT.9.0R. TEMPX(1).LT.0) GO TO 15
      ISIG(KUP)=TEMPX(1)
      GO TO 5050
C
C
      5025 CONTINUE
C
      C.....FREQUENCY IN HZ
C
      IF(TEMPX(1).GT.72.0R. TEMPX(1).LT.1) GO TO 15
      ISIG(KUP)=TEMPX(1)
      GO TO 5050
C
      5026 CONTINUE
C
      C.....TIME IN MILLISECONDS
C
      IF(TEMPX(1).LT.10.0R. TEMPX(1).GT.9999) GO TO 15
      R=TEMPX(1)*1.6384
      T16=INUM(R,1)
      ISIG(KUP)=T16*512+ICHAN(I+11)
      GO TO 5050
C

```



```

5028 CONTINUE
C
C.....TIME IN MILLISECONDS
C
      IF(TEMPX(1).LT.10.0R,TEMPX(1),GT,4090) GO TO 15
      ISIG(KUP)=TEMPX(1)*2**11+00003777B
C
5050 KUP=KUP-1
      ENCODE(16,1700,TEMPY)TEMPX(1)
      CALL TEXT0(IDEV,TEMPY,4,3*I+5,81,1,2,IER)
1700 FORMAT(6X,$= $,1I7,$,$)
100 CONTINUE
C
1711 FORMAT(8X,1I8)
C
101 CONTINUE
      ISUM=0
C
      DO 102 I=1,I8
102  ISUM=ISUM+IBIT(I)*2**(I-1)
C
      ISIG(KDEF)=ISUM+KSTRT*4096
C
      DO 200 J=1,I8
      IF(ISUM.EQ.0) GO TO 104
      CALL TEXT1(IDEV,TEMPX,4,3*J+5,81,IER)
      CALL TEXT0(IDEV,TEMPX,4,3*J+5,1,1,2,IER)
104  CONTINUE
      CALL TEXT0(IDEV,NULL,4,3*J+5,81,1,2,IER)
200 CONTINUE
C
      IF(ISUM.EQ.0) GO TO 1
      KDEF=KSH0=KDEF+1
      KSTRT=KUP

```



```

LOGICS(32)=1
IF(KOKD.EQ.1) GO TO 1001
GO TO 1

C
1000 CONTINUE
IBTOP=KDEF
ITTOP=KUP
1001 CONTINUE
CALL SHSIG(2,-1,-1,1,1,18)
CALL SHSIG(2,1,-1,1,1,18)
CALL BUTIN(-1,NND,10,6,7,8,9,10)
CALL TEXT0(IDEV,NULL,5,3,1,1,1,IER)
CALL TEXT0(IDEV,NULL,5,3,7,7,1,1,IER)

C
RETURN
3000 CONTINUE
NX=KDEF-1
KDIV=ISIG(NX)
KDL9C=KDIV/2*12
CALL BITREC(NX,IXH9LD,ISUM)
KSTRT=KDL9C-ISUM
GO TO 3001
END

```



```

C
C SUBROUTINE SHSIG(NX,LR,IV3,NNB, NNT)
C
C SUBROUTINE TO DISPLAY A PREVIOUSLY DEFINED SIGNAL OR LIST THE
C ELEMENTS TO BE DEFINED AND TO ERASE THE ABOVE DISPLAYS.
C ALSO USED TO DISPLAY TEXT IN MATRIX NOTES.
C NX=THE NUMBER OF THE SIGNAL TO BE DISPLAYED.
C LR 0=DISPLAY ON RIGHT HALF OF THE CRT.
C 1= DISPLAY ON LEFT HALF.
C NNB=LOWEST VECTOR IN NOTES TO BE DISPLAYED.
C NNT=HIGHEST.
C
C INTEGER ASIG,BSIG,AMP,FREQ
C INTEGER BUTTON,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
C INTEGER TEMPSL,TEMPBP
C COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
C COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
C COMMON KFER,KAMP,KS2,KSIG,I8,I8TOP,ITT9P
C COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
C * FREQ(72),KAAK,KDIV,KUP
C COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
C COMMON/KDS/KDISP(8,0:9,3)
C COMMON/IGRAF/IBOX(8),IGRID(35),BUTTON(2),IFIG(100)
C COMMON IWORKA(10),IXHOLD(15),TEMPX(10),TEMPY(10),
C * TEMPDX(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
C COMMON KEUTIN(15),IWORKB(15)
C COMMON/INFO/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
C * IDC1,IDF1
C
C C.....IVB=-1: BLANK DATA
C
C C.....IVB=0 TITLES ONLY
C
C C.....IVB=1 TITLES AND DATA
C

```



```

C      KLR=1
      IF(LR.GE.0) KLR=81
C
C      IF(IVB)1000,2,2
C
      2 CONTINUE
      K=0
      DO 100 I=NNB,NNT
      K=K+1
      CALL TEXT8(IDEV,NOTES(1,I),4,3*K+4,KLR,1,2,IER)
100 CONTINUE
C
      IF(IVB.EQ.0) RETURN
C
      DO 1 I=1,12
      1 IBIT(I)=0
C
      L8C=ISIG(NX)/4096
C
      CALL BITREC(NX,IWRKA,ISUM)
C
      DO 200 I=1,I8
      IF(IWRKA(I).EQ.0) GO TO 200
      IDD=ISIG(L8C)
      IF(I.LT.6.8R.I.GT.7) GO TO 190
      T26=IDD/512
      R=T26/1.6384
      IDD=INUM(R,2)
190 CONTINUE
      IF(I.EQ.8)IDD=IDD/2**11
      ENCODE(16,1700,TEMPX)IDD
1700 FORMAT(6X,5= $,1I7,$,$)
      L8C=L8C-1
      CALL TEXT8(IDEV,TEMPX,4,3*I+5,KLR,1,2,IER)

```



```

200 CONTINUE
C
IF(ISUM.EQ.18)RETURN
KS2=ISUM
C
DO 210 I=1,18
IF(IW0RKA(I).EQ.1) GO TO 210
K=NX-1
203 CALL BITREC(K,IW0RKB,ISUM)
IF(IW0RKB(I).EQ.1) GO TO 205
K=K-1
GO TO 203
C
205 CONTINUE
L0C=ISIG(K)/4096
ISUM=0
C
DO 206 J=1,I
206 ISUM=ISUM+IW0RKB(J)
C
IDD=ISIG(L0C-ISUM+1)
IF(I.LT.6.OR.I.GT.7) GO TO 207
T26=IDD/512
R=T26/1.6384
IDD=INUM(R,2)
207 CONTINUE
IF(I.EQ.8)IDD=IDD/2**11
ENCODE(16,1701,TEMPX)IDD
CALL TEXT0(IDEV,TEMPX,4,3*I+5,KLR,1,2,IER)
1701 FORMAT(3X,$*,2X,$= $,117,$,$)
210 CONTINUE
C
RETURN
C
C

```



```
1000 CONTINUE
      DO 1010 I=1,9
      CALL TEXT0(IDEV,NULL,4,3*I+4,KLR,1,2,IER)
      CALL TEXT0(IDEV,NULL,4,3*I+5,KLR,1,2,IER)
1010 CONTINUE
      C
      RETURN
      END
```



```

C
C SUBROUTINE DEFINEN(NR,MODE,KDKD,/KSKIP(KSK)/)
C
C SUBROUTINE USED TO DEFINE ONE OF THE FUNCTIONS.
C
C NR=THE NUMBER OF THE ELEMENT.
C MODE: 1= PERIODIC WAVE FUNCTION.
C        2=AM FUNCTION
C        3=FM FUNCTION.
C
C IF THE NUMBER OF ELEMENTS IN KSK IS GREATER THAN ZERO,
C THE MAGNITUDE OF THE FUNCTION IS NORMALIZED SO THAT THE
C LARGEST VALUE IS. 1000 UNITS.
C
C INTEGER ASIG,BSIG,AMP,FREQ
C INTEGER BUTTEN,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
C INTEGER TEMPSL,TEMPBP
C COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
C COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
C COMMON KFRE,KAMP,KS2,KSIG,I8,IBT0P,ITT9P
C COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
C        * FREQ(72),KAAK,KDIV,KUP
C COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
C
C COMMON/KDISP(8,0:9,3)
C COMMON/IGRAF/IBOX(8),IGRID(35),BUTTEN(2),KFIG(100)
C COMMON IWRKA(10),IXH9LD(15),TEMPX(10),TEMPY(10),
C        * TEMPDX(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
C COMMON KAUTIN(15),IWRKB(15)
C COMMON/INF9/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
C        * IDC1,IDF1
C
C DIMENSION IFIG(400)
C DATA IFIG/400*0/
C
C.....IWAY=1 : GRAPHICAL INPUT      IWAY=2 : TYPEWRITTEN

```



```

C.....MODE : 1.WAVE, 2.AMP FUNCT, 3.FREQ FUNCT
C
C
1 CONTINUE
IWAY=1
CALL GRAPH9(IDEV,IBOX,8,1,IER)
CALL GRAPH9(IDEV,IGRID,35,2,IER)
3222 ITD=1
C
2 CONTINUE
C
ENCODE(48,1720,TXT(1,9))
ENCODE(48,1721,TXT(1,10))
1720 FORMAT($ TYPE L FOR LIGHT PEN INPUT,
1721 FORMAT($ T FOR TYPEWRITE INPUT, C/R FOR RETURN
$)
$)
CALL OUTIN(0,NND,10,9,10)
K23=2
IFIG(1)=KHED(1)
IFIG(2)=KHED(2)
IF(MODE.NE.2) GO TO 6002
CALL UNPACK(KDISP(8,NR,2),TEMPX(1),TEMPY(1),IMD)
IFIG(3)=IPACK(-1000,TEMPY(1),1)
K23=3
6002 CONTINUE
C
3 CONTINUE
C
K=1
DO 4 I=K23+1,K23+8
IFIG(I)=KDISP(K,NR,MODE)
K=K+1
4 CONTINUE
C
IF(ITD.EQ.2) GO TO 1005
C

```



```

K23=K23+9
IFIG(K23)=0
IFIG(K23+1)=0
K23=K23+1
9795 CONTINUE
      6 CONTINUE
      CALL GRAPH0(IDEV,FIG,K23,3,IER)
9797 CALL BUTIN(1,NND,8,9,10)
      IF(NND.EQ.3) GO TO 1006
      IF(KBUTIN(1).NE.4HT ) GO TO 9794
      CALL BUTIN(-1,NND,8,8,9,10)
      GO TO 500
9794 CONTINUE
      IF(KBUTIN(1).NE.4HL ) GO TO 9797
      CALL BUTIN(-1,NND,8,8,9,10)
      CALL GRAPHR(IDEV,FIG,K23,3,IER)
      C
      C
      C
      C
      10 IF(M0D(IGDIR(3),8).EQ.0) GO TO 10
      CALL GRAPHI(IDEV,FIG,3,IER)
      C
      D0 5678 I=1,15
5678 IXHOLD(I)=FIG(I)
      C
      C
      C
      MAXY=0
      C
      11 CONTINUE
      CALL BUTIN(-1,NND,10,6,10)
      K=0
      K1=2
      IF(M0DE.EQ.2) K1=J
      K2=K1+8

```



```

C
IF(IWAY.EQ.2) GO TO 13

DO 30 I=K1,K2
K=K+1
CALL UNPACK(IXHOLD(I),TEMPX(K),TEMPY(K),IMD)
IF(K.NE.9) GO TO 22
TEMPY(9)=TEMPY(1)
22 CONTINUE
IF(K.NE.1) GO TO 25
TEMPX(1)=-1000
IF(MODE.NE.2) TEMPY(1)=0
25 CONTINUE
IBIG=IABS(TEMPY(K))
IF(1BIG.GT.MAXY)MAXY=1BIG
30 CONTINUE
13 CONTINUE
XJUST=2000.0/(TEMPX(9)+1000.0)

C
DO 21 I=1,9
TEMPX(I)=(TEMPX(I)+1000.0)*XJUST-1000.0
21 CONTINUE

C
IF(KSK.EQ.0.AND.MAXY.LE.1000)GO TO 131

C
AJUST=1000.0/MAXY

C
DO 31 I=1,9
TEMPY(I)=TEMPY(I)*AJUST
31 CONTINUE

C
131 CONTINUE

C
C.....CHECK FOR ZERO OR NEGATIVE DELTA X
C
DO 34 I=1,8

```



```

TEMPDX(I)=TEMPX(I+1)-TEMPX(I)
IF(TEMPDX(I).GT.0) GO TO 32
GO TO 3222

C
32 CONTINUE
TEMPDY(I)=TEMPY(I+1)-TEMPY(I)
TEMPSL(I)=TEMPDY(I)/(TEMPDX(I)+0.0)*SLFAC(MODE)
IF(NR.EQ.0) GO TO 34
IF(ABS(TEMPSL(I)).LE.SLMAX(MODE)) GO TO 33
GO TO 3222
33 CONTINUE
IF(MODE.EQ.1.AND.I.EQ.8.AND.ABS(TEMPSL(8)).LT.SLMIN(2)) GO TO 3222
IF(ABS(TEMPSL(I)).GE.SLMIN(MODE)) GO TO 34
GO TO 3222
34 CONTINUE

C
35 CONTINUE
DO 40 I=1,8
C
C
C
37 CONTINUE

T16=(TEMPSL(I)/10000.1)*2**14
T16=T16*2**9
IF(MODE.NE.1) GO TO 38
T26=((TEMPX(I+1)+1000.0)*BPFAC(1)/20000.1)*2**14
GO TO 39
38 CONTINUE
T26=(TEMPY(I+1) *BPFAC(MODE)/10000.1)*2**14
39 CONTINUE
T26=T26*2**9
T36=ASIG(I,NR,MODE)
T36=T36-(T36/512)*512
ASIG(I,NR,MODE)=T36+T16
T36=BSIG(I,NR,MODE)

```



```

T36=T36-(T36/512)*512
BSIG(I,NR,MODE)=T36+T26
40 CONTINUE
C
D0 45 I=1,8
KDISP(I,NR,MODE)=IPACK(TEMPX(I+1),TEMPY(I+1),1)
45 CONTINUE
C
ITD=2
G0 T0 2
C
500 CONTINUE
IWAY=2
K1=1
IF(MODE.EQ.2)K1=2
IBIT(1)=0
C
D0 511 I=1,8
CALL UNPACK(KDISP(I,NR,MODE),TEMPX(I+1),TEMPY(I+1),IMD)
IBIT(I+1)=(TEMPX(I+1)+1000)/2
C
511 CONTINUE
TEMPX(1)=-1000
TEMPY(1)=TEMPY(9)
TEMPX(9)=1000
C
C.....DISPLAY X AND Y COORDINATES OF FUNCTION OR WAVE
C
D0 510 I=1,9
ENCODE(16,1700,TXT(1,6))I
ENCODE(16,1701,TXT(1,7))I
1700 FORMAT($X($,111,$) = $,8X)
1701 FORMAT(8X,$Y($,111,$) = $)
CALL TEXT0(IDEV,TXT(1,6),4,3*I+4,1,1,2,IER)
CALL TEXT0(IDEV,TXT(1,7),4,3*I+4,81,1,2,IER)

```



```

C      ENCODE(16,1702,TXT(1,6))IBIT(I)
      ENCODE(16,1712,TXT(1,7)) TEMPY(I)

C      1702 FORMAT(118,8X)
      1712 FORMAT(8X,118)
      CALL TEXT0(IDEV,TXT(1,6),4,3*I+5,1,1,2,IER)
      CALL TEXT0(IDEV,TXT(1,7),4,3*I+5,81,1,2,IER)
      510 CONTINUE
      IBIT(1)=0

C      DO 550 I=1,9
      IFLAG=1
      IGZ=I
      IF(I.EQ.1)GO TO 42

C      ENCODE(48,1703,TXT(1,8))I,I-1,I
      1703 FORMAT($INPUT X($,111,$)  + X($,111,$) +1000$,16X)
      1613 CONTINUE
      CALL OUTIN(1,NND,10,8)

C      GO TO (512,513)IFLAG
      512 CONTINUE
      GO TO (514,1613,1517,555,1613)NND
      513 CONTINUE
      GO TO (514,1613,550,555,1613)NND

C      514 CONTINUE
      DECODE(8,1702,KOUTIN)IWORKA(I)
      IF(IFLAG.EQ.1) IBIT(I)=IWORKA(I)
      3514 CONTINUE
      IF(IFLAG.EQ.2) GO TO 525
      IF(IWORKA(I).LE.1000.AND.IWORKA(I).GT.IBIT(I-1)) GO TO 517

C      516 CONTINUE
      GO TO 1613

```



```

C 517 CONTINUE
    ENCODE(16,1702,IXHOLD)IWORKA(I)
    TEMPX(I)=IWORKA(I)*2-1000
    CALL TEXT0(IDEV,IXHOLD,4,3*I+5,1,1,2,IER)

C 1517 CONTINUE
    GO TO 42
1518 CONTINUE

C          IFLAG=2
          IF(MODE.NE.2.AND.I.EQ.1) GO TO 550
          ENCODE(48,1704,TEXT(1,8))I,I
1704 FORMAT($INPUT Y($,1I1,$). -1000 + Y($,1I1,$) + 1000$,16X)
520 CONTINUE
    GO TO 1613

C 525 CONTINUE
    IF(ABS(IWORKA(I)).LE.1000) GO TO 526
    GO TO 1613

C 526 CONTINUE
    TEMPY(I)=IWORKA(I)
    ENCODE(16,1712,IXHOLD)IWORKA(I)
    CALL TEXT0(IDEV,IXHOLD,4,3*I+5,81,1,2,IER)
    GO TO 42
550 CONTINUE
555 CONTINUE

C      TEMPY(9)=TEMPY(1)
C      IF(MODE.NE.1) GO TO 11
C      MAXY=0

```



```

C      DO 560 I=1,9
        IBIG=IABS(TEMPY(I))
        IF(IBIG.GT.MAXY)MAXY=IBIG
560    CONTINUE
        GO TO 11
C
C
C      1000 CONTINUE
        GO TO 2
C
C      1005 CONTINUE
        CALL BUTIN(-1,NND,10,8)
        IFIG(12)=IFIG(13)=IFIG(14)=0
        CALL GRAPH9(IDEV,IFIG,14,3,IER)
        1706 FORMAT($ TYPE END IF DEFINITION COMPLETE $)
C
C      ENCODE(48,1706,TXT(1,9))
        CALL BUTIN(1,NND,10,9)
        IF(NND.EQ.5) GO TO 1006
        ITD=1
        GO TO 2
C
C.....BLANK TEXT AND GRAPHICS
C
C      1006 CONTINUE
C
C      DO 1010 I=1,50
        IFIG(I)=0
1010    CONTINUE
C
        CALL GRAPH9(IDEV,IFIG,8,1,IER)
        CALL GRAPH9(IDEV,IFIG,35,2,IER)
        CALL GRAPH9(IDEV,IFIG,14,3,IER)
        CALL BUTIN(-1,NND,10,6,7,8,9,10)

```



```

C      CALL SHSIG(0,1,-1,0,0)
C      CALL SHSIG(0,-1,-1,0,0)
C      RETURN
C
C      42 CONTINUE
C      IFIG(2)=IPACK(TEMPX(1),TEMPY(1),0)
C      LL=1
C
C      DO 46 L=3,11
C      LL=LL+1
C      IFIG(L)=IPACK(TEMPX(LL),TEMPY(LL),1)
C      46 CONTINUE
C
C      IFIG(11)=IFIG(12)=IFIG(13)=0
C      CALL GRAPH8(IDEV,IFIG,12,3,IER)
C      GO TO (1518,550)IFLAG
C      END

```



```

C
C SUBROUTINE BITREC(NX, IVEC, ITOT, /KD(NN)/)
C
C SUBROUTINE TO DECODE THE INFORMATION IN ARRAY ISIG ABOUT A SIGNAL.
C NX=MAIN SIGNAL SEQUENCE NUMBER OF SIGNAL TO BE DECODED,
C IVEC=VECTOR IN WHICH INFORMATION ABOUT SIGNAL IS RETURNED.
C ITOT=NUMBER OF ELEMENT TYPES DEFINED FOR THE SIGNAL
C KD: IF SIZE OF VECTOR KD IS GREATER THAN ZERO, TRANSFER
C IS MADE TO SUBROUTINE COMMENT.
C
C INTEGER ASIG,BSIG,AMP,FREQ
C INTEGER BUTTON,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
C INTEGER TEMPSL,TEMPBP
C COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
C COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
C COMMON KFER,KAMP,KS2,KSIG,I8,I8T9P,ITT9P
C COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
C * FREQ(72),KAAK,KDIV,KUP
C COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
C COMMON/KDS/KDISP(8,0:9,3)
C COMMON/IGRAF/IBEX(8),IGRID(35),BUTTON(2),IFIG(100)
C COMMON IWARKA(10),IXH9LD(15),TEMPX(10),TEMPY(10),
C * TEMPDX(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
C COMMON KOUTIN(15),IWARKB(15)
C COMMON/INFO/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
C * IDC1,IDF1
C DIMENSION IVEC(12)
C
C IF(NN.EQ.1) GO TO 1001
C
C DO 1 I=1,12
C 1 IVEC(I)=0
C ITOT=0
C
C KDIV=ISIG(NX)
C

```



```

D0 100 I=1,18
KAAK=KDIV
KDIV=KAAK/2
IVEC(I)=KAAK-2*KDIV
IF(IVEC(I).EQ.1) ITOT=ITOT+1
100 CONTINUE
C
RETURN
1001 CONTINUE
CALL SHSIG(0,1,-1,15,23)
CALL SHSIG(0,-1,-1,1,2)
CALL SUTIN(-1,NND,10,6,7,8,9,10)
CALL COMMENT
RETURN
C
SUBROUTINE COMMENT
SUBROUTINE TO TRANSFER COMMENTS ENTERED ON AGT TELETYPEWRITER
TO LINE PRINTER.
C
ENCODE(48,800,TXT(1,9))
800 FORMAT($ TYPE COMMENTS OR END IF FINISHED $)
C
1000 CONTINUE
CALL SUTIN(0,NND,10,9)
C
CALL TEXTR(IDEV,NULL,1,25,1,1,2,IER)
2005 IF(MOD(ITDIR(6),8).EQ.0) GO TO 2005
C
CALL TEXTI(IDEV,IFIG,24,25,1,IER)
IF(IFIG(1).EQ.4HEND ) GO TO 999
IF(IFIG(1).NE.4HTOP ) GO TO 3000
WRITE(6,802)

```



```

802 FORMAT($1$)
GO TO 1000
3000 IF(IFIG(1).NE.4HSTAR) GO TO 3500
WRITE(6,805)
805 FORMAT($0$,10X,24(4H****))
GO TO 1000
3500 CONTINUE
WRITE(6,803)(IFIG(I),I=1,24)
803 FORMAT($$,10X,24A4)
GO TO 1000

C C
999 CONTINUE
CALL OUTIN(-1,NND,10,9)
CALL TEXT0(IDEV,NULL,1,25,1,1,2,IER)

C C
C RETURN
C END

```



```

1 CONTINUE
C
C.....PRINT AMP
C
WRITE(6,801)
C
DO 100 I=0,9
T16=AMP(I)/2**9
R=T16/1.6384
T16=INUM(R,2)
WRITE(6,800)NOTES(1,4),I,T16
C
800 FORMAT($0$,10X,1A4,$($,1I2,$) = $,1I4)
100 CONTINUE
C
801 FORMAT($1$,////)
RETURN
2 CONTINUE
C.....PRINT FREQ
C
WRITE(6,801)
C
DO 200 I=1,72
T16=FREQ(I)/2**9
R=T16/1.6384
T16=INUM(R,2)
IA(I)=T26=T16/10
IB(I)=T16-10*T26
200 CONTINUE
C
DO 300 I=1,24
WRITE(6,815)(J,IA(J),IB(J),J=I,I+48,24)
300 CONTINUE
815 FORMAT($0$,10X,$,1I2,$) = $,1I3,$. $,1I1,13X))

```



```

C      RETURN
C      3 CONTINUE
C      C.....PRINT SIGNAL SEQUENCE
C      803 FORMAT($1$,20A4,2(4A4,$/$),4A4,/)
C      KUP=1001
C      ENCODE(48,806,TXT(1,6))
C      806 FORMAT($ INPUT FIRST NUMBER OF SEQUENCE $)
C      ENCODE(48,807,TXT(1,7))
C      807 FORMAT($ USE C/R FOR CURRENT LIMITS $)
C      808 IWUN=1
C      CALL GUTIN(1,NND,10,6,7)
C      IF(NND.EQ.5) GO TO 501
C      IF(NND.EQ.3) GO TO 811
C      IF(NND.NE.1) GO TO 808
C      DECODE(4,805,KEUTIN)IFIG(1)
C      IF(IFIG(1).LT.1.OR.IFIG(1).GT.IBTOP-1) GO TO 808
C      IWUN=IFIG(1)
C      811 CONTINUE
C      ENCODE(48,809,TXT(1,6))
C      809 FORMAT($ INPUT LAST NUMBER OF SEQUENCE $)
C      810 I2=IBTOP-1
C      CALL GUTIN(1,NND,10,6)
C      IF(NND.EQ.5) GO TO 501
C      IF(NND.EQ.3) GO TO 812
C      IF(NND.NE.1) GO TO 810
C      DECODE(4,805,KEUTIN)IFIG(2)
C      IF(IFIG(2).LT.IFIG(1).OR.IFIG(2).GT.IBTOP-1) GO TO 810
C      I2=IFIG(2)
C      812 CONTINUE
C

```



```

C      WRITE(6,803)((NOTES(I,J),I=1,4),J=1,8)
C
C      DO 500 I=1,IBTOP-1
C      CALL BITREC(I,IBIT,ISUM)
C
C      DO 450 J=1,8
C      TEMPY(J)=4H *
C      IF (IBIT(J).EQ.0) GO TO 450
C      KUP=KUP-1
C      IF (J.LE.5) TEMPX(J)=ISIG(KUP)
C      IF (J.LT.6.0R.J.GT.7) GO TO 440
C      T16=ISIG(KUP)/2**9
C      R=T16/1.6384
C      TEMPX(J)=INUM(R,2)
C      GO TO 449
C
C      440 IF (J.EQ.8) TEMPX(8)=ISIG(KUP)/2**11
C      449 ENCODE(4,805,TEMPY(J))TEMPX(J)
C      450 CONTINUE
C      805 FORMAT(1I4)
C      IF (I.LT.IWUN.0R.I.GT.12) GO TO 500
C      WRITE(6,804)I,(TEMPY(J),J=1,8)
C      804 FORMAT($0$,1I4,4X,1A4,4X,7(6X,1A4,6X))
C      500 CONTINUE
C      501 CONTINUE
C      CALL 9UTIN(-1,NND,10,6,7,10)
C      RETURN
C
C      888 CONTINUE
C
C      DO 1111 I=1,72
C      T16=FREQ(I)/2**9
C      R=T16/1.6384
C      T16=INUM(R,2)
C      IA(I)=T16/10

```



```

IB(I)=T16-10*IA(I)
1111 CONTINUE
C
1117 CONTINUE
C
K=0
DO 2222 I=1,17,2
K=K+1
KKK=I+55
ENCODE(96,1801,IFIG(75))(J,IA(J),IB(J),J=1,KKK,18)
1801 FORMAT(4($F(5,112,$) = $,113,$.5,111,6X))
CALL TEXT0(IDEV,IFIG(75),24,3*K+4,1,1,2,IER)
ENCODE(96,1801,IFIG(75))(J,IA(J),IB(J),J=I+1,KKK+1,18)
CALL TEXT0(IDEV,IFIG(75),24,3*K+5,1,1,2,IER)
2222 CONTINUE
C
C
C
C
C
SELECT FREQUENCY ELEMENT
C
2000 CONTINUE
ENCODE(48,1803,TXT(1,7))
3000 CALL SUBIN(1,NND,10,7)
IF(NND.EQ.5) GO TO 999
IF(NND.NE.1) GO TO 3000
DECODE(4,805,KSUBIN)T16
IF(T16.LT.1.0R.T16.GT.72) GO TO 3000
C
C
C
C
INPUT FREQUENCY OF ELEMENT
C
4000 ENCODE(48,1804,TXT(1,7))
CALL SUBIN(1,NND,10,7)
IF(NND.EQ.5) GO TO 999
IF(NND.NE.1) GO TO 4000

```



```

DECODE(4,805,KOUTIN)T26
IF(T26.LT.100.GR.T26.GT.9999) G0 T0 4000
IF(KD(1).EQ.4H9CTA) G0 T0 7000
IF(KD(1).EQ.4HSTEP)G0 T0 8000
R=T26*1.6384
T36=INUM(R,1)
FREQ(T16)=T36*2**9+ICHAN(14)
IA(T16)=T26/10
IB(T16)=T26-10*IA(T16)
T26=MOD(T16-1,18)+1
KKK=T26+54
IK=5
T36=T26/2
IF(2*T36.EQ.T26) G0 T0 4010
IK=4
T36=T36+1
4010 CONTINUE
ENCODE(96,1801,IFIG(75))(J,IA(J),IB(J),J=T26,KKK,18)
CALL TEXT0(IDEV,IFIG(75),24,3*T36+IK,1,1,3,IER)
G0 T0 2000

C
C
999 CONTINUE
CALL BUTIN(-1,NND,10,6,7,10)
D0 9000 I=60,84
9000 IFIG(I)=NULL(1)

C
D0 9002 I=1,9
CALL TEXT0(IDEV,IFIG(60),24,3*I+5,1,1,1,IER)
9002 CALL TEXT0(IDEV,IFIG(60),24,3*I+4,1,1,1,IER)
1803 FORMAT($ INPUT FREQUENCY ELEMENT NUMBER $)
1804 FORMAT($ INPUT FREQUENCY X 10, 100 + INPUT + 9999 $)

C
RETURN
7000 CONTINUE

```



```

K=0
ENCODE(48,1806,TXT(1,7))
1806 FORMAT($ INPUT EXPONENT NUMERATOR $)
CALL SUBIN(1,NND,10,7)
IF(NND.EQ.5) GO TO 999
IF(NND.NE.1) GO TO 7000
DECODE(4,805,KOUTIN)KPART
IF(KPART.LT.0.9R.KPART.GT.100) GO TO 7000
C
DO 7010 I=T16,T16+KPART
NND=T26*2*(K/(KPART+0.0))
IF(I.LT.1.8R.I.GT.72.8R.NND.GT.9999.8R.NND.LT.100)GO TO 7010
R=NND*1.6384
FREQ(I)=INUM(R,1)
FREQ(I)=FREQ(I)*2**9+ICHAN(14)
IA(I)=NND/10
IB(I)=NND-10*IA(I)
K=K+1
7010 CONTINUE
GO TO 1117
C
C
8000 CONTINUE
K=0
NNN=T16+9
IF(KPART.LT.0)NNN=T16-9
C
ENCODE(48,1805,TXT(1,7))
CALL SUBIN(1,NND,10,7)
1805 FORMAT($ INPUT INCREMENT X 10, 10+INPUT+1000 $)
IF(NND.EQ.5) GO TO 999
IF(NND.NE.1) GO TO 8000
DECODE(4,805,KOUTIN)KPART
IF(KPART.LT.10.9R.KPART.GT.1000) GO TO 8000
C

```



```

D8 8010 I=T16,NNN
NND=T26+K
IF(I.LT.1.8R.I.GT.72.8R.NND.GT.9999.8R.NND.LT.100)G8 T8 8010
C
R=NND*1.6384
FREQ(I)=INUM(R,1)
FREQ(I)=FREQ(I)*2**9+ICHAN(14)
IA(I)=NND/10
IB(I)=NND-10*IA(I)
K=K+KPART
8010 CONTINUE
C
G8 T8 1117
C
END

```



```

C C C C C C C C C C C C C C C C
SUBROUTINE FRESPEC(NR1,NR2,IWAY,RAT,MODE,KD,KPART1,KPART2,LPT)

SUBROUTINE USED TO CONTROL FUNCTION SAMPLING AND FREQUENCY
ANALYSIS. ALSO USED TO CONTROL PLOTTING OF FUNCTIONS.

NR1=NUMBER OF FIRST FUNCTION
NR2=NUMBER OF SECOND FUNCTION (IF PRODUCT OF TWO FUNCTIONS).
IWAY: =1 IF SINGLE FUNCTION
      =2 IF PRODUCT OF TWO FUNCTIONS
RAT=RATIO OF PERIODS OF FIRST FUNCTION TO SECOND
KD: 0= PLOT FUNCTIONS
    1= FIND SPECTRUM
MODE: 1=PERIODIC WAVE
      2=AM FUNCTION
      3=FM FUNCTION
KPART1: KPART1/1000=AMOUNT OF FIRST FUNCTION TO BE USED
KPART2: KPART2/1000=AMOUNT OF SECOND (OR FIRST IF ONLY
ONE FUNCTION) NOT SET EQUAL TO ZERO
LPT=NUMBER OF SAMPLES IN TRANSFORM OR PLOTS

INTEGER ASIG,BSIG,AMP,FREQ
INTEGER BUTTN,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
INTEGER TEMPSL,TEMPBP
COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
COMMON KFRE,KAMP,KS2,KSIG,I8,IBTOP,ITTP
COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
* FREQ(72),KAAK,KDIV,KUP
COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
COMMON/KDS/KDISP(8,0:9,3)
COMMON/IGRAF/IBOX(8),IGRID(35),BUTTN(2),IXX(100)
COMMON IWORKA(10),IXHOLD(15),TEMPX(10),TEMPY(10),
* TEMPD(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
COMMON KOUTIN(15),IWORKB(15)
COMMON/INFO/NOTES(10),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,

```



```

C
C
* IDC1,IDF1

COMPLEX CDT
DOUBLE PRECISION BIG, AJUST
DIMENSION Y(2048),CDT(1024),IFIG(800),XY(400),XXYY(400)
DIMENSION ITX(400)
EQUIVALENCE(IFIG(401),ITX(1))
EQUIVALENCE(IFIG(1),XXYY(1))
EQUIVALENCE(Y(1),CDT(1))

C
C
NPT=LPT
IF(IWAY.EQ.2)NPT=2*LPT
IF(SENSESWITCH 1)205,206
205 OUTPUT(6)NR1,NR2,IWAY,RAT,KD,NPT,KPART1,KPART2,MODE
PAUSE
206 CONTINUE
CALL FSAMP(NR1,NR2,IWAY,RAT,KD,NPT,KPART1,KPART2,Y,MODE)
IF(KD.EQ.0) GO TO 3000

C
C
K=2*NPT-1

C
C
CALL FOURZ(Y,NPT,1,-1)

C
C
999 CONTINUE
135 CONTINUE

C
ENCDE(48,9000,TXT(1,7))
9000 FFORMAT($ H8W MANY POINTS SHALL BE DISPLAYED $)
ENCDE(48,9001,TXT(1,8))
9001 FFORMAT($ MINIMUM = 5 MAXIMUM = 399 $)
ENCDE(48,9003,TXT(1,9))

```



```

9003 FORMAT($                                $)
180 CALL DUTIN(1,NND,10,7,8,9)
C
C
IF(NND.EQ.5) GO TO 191
IF(NND.NE.1) GO TO 180
C
DEC9DE(4,9002,K8UTIN)KNP
9002 FORMAT(114)
C
190 CONTINUE
IF(KNP.LT.5.E8.KNP.GT.399) GO TO 180
CALL DUTIN(-1,T16,10,7,8,9,10)
KDELT=2000/(KNP-1)
C
C
191 CONTINUE
C
DO 600 I=1,800
IFIG(I)=0
600 CONTINUE
C
CALL GRAPH9(IDEV,IFIG,400,3,IER)
CALL GRAPH8(IDEV,IFIG,400,4,IER)
CALL GRAPH8(IDEV,IFIG,9,5,IER)
C
IF(NND.EQ.5) GO TO 500
K=-1
BIG=0.0
C
DO 200 I=1,KNP
K=K+2
R=XY(I+1)=CABS(CDT(I))
IF(R.GT.BIG)BIG=R
AT=ATAN(Y(K+1)),

```



```

IF(SENSESWITCH 5)7766,7767
7766 WRITE(6,867)I-1,CDT(I),R,AT
867 FORMAT($ F($,1I3,$) = $,1F10.5,$ + J $,1F10.5,$ = $,1F10.5,
* $ ($,1F10.5,$ RADIANS)$)
7767 CONTINUE
ITX(I+1)=AT*127.32395-600.0
200 CONTINUE
C
C
AJUST=LOGICS(25)
IF(SENSESWITCH 4) 202,201
201 CONTINUE
AJUST=1000.0/BIG
202 CONTINUE
LOGICS(25)=AJUST
K=1000
C
C
IF(AJUST.LE.0.0) AJUST=1.0
D8 220 I=2,KNP+1
IFIG(I)=XY(I)*AJUST
IFIG(I)=IPACK(K,IFIG(I),1)
ITX(I)=IPACK(K,ITX(I),1)
K=K+KDELT
220 CONTINUE
C
C
ITX(1)=IFIG(1)=IHEAD(0,8)
IFIG(2)=IFIG(2)-00000001B
ITX(2)=ITX(2)-00000001B
C
C
CALL GRAPH9(IDEV,LOGICT,9,5,IER)
CALL GRAPH9(IDEV,IFIG,KNP+1,3,IER)
CALL GRAPH9(IDEV,,1,4,IER)

```



```

9006 ENCODE(48,9006, TXT(1,8))
      FORMAT($ FOR SPECTRUM PLOT, TYPE P $)
      CALL DUTIN(1,NND,10,8)
      IF(IWORKB(1).EQ.4H000P) GO TO 1500
227 CONTINUE
      GO TO 135
C
500 CONTINUE
C
      CALL DUTIN(-1,NND,10,7,8,9,10)
C
C
      RETURN
1500 CONTINUE
      IF(KNP.GT.200)KNP=200
C
C
      DO 1502 I=1,KNP
        XYY(I+KNP)=XY(I+1)*AJUST
        XYY(I)=I-1
1502 CONTINUE
C
      TEMPX(1)=1
      TEMPX(2)=2
      805 FORMAT($ $,10X,$FREQUENCY SPECTRUM MAGNITUDE - $,
        * $ NUMBER OF POINTS = $,114)
        IF(IWAY.EQ.1) WRITE(6,806)(N0TES(I,M0DE),I=1,4),NR1
        IF(IWAY.EQ.2) WRITE(6,807)(N0TES(I,1),I=1,4),NR1,
        * (N0TES(J,2),J=1,4),NR2,RAT
        WRITE(6,805)KNP
C
      807 FORMAT($1$,10X,4A4,2X,$NUMBER$,2X,11,$ X $,4A4,2X,
        $ $NUMBER$,2X,11,5X,$RATIO = $,1F5,2)
        PNK=KNP-1
        CALL ZPLOT(XYY, , ,KNP,-1,1,0.0,PNK,0.0,1000.0)

```



```

C
C      CALL DUTIN(-1,NND,10,8,10)
C      GO TO 135
C
C      3000 CONTINUE
C
C      DO 3001 I=2,NPT
C      Y(I)=Y(2*I-1)
C      3001 CONTINUE
C
C      K=0
C      DD=1000.0/(NPT-1)
C
C      DO 3010 I=NPT+1,2*NPT
C      Y(I)=K*DD
C      K=K+1
C      3010 CONTINUE
C
C      TEMPX(1)=2
C      TEMPX(2)=1
C      IF(IWAY.EQ.1) GO TO 823
C      WRITE(6,807)(NOTES(I,1),I=1,4),NR1,(NOTES(J,2),J=1,4),NR2,RAT
C      GO TO 824
C      823 CONTINUE
C
C      WRITE(6,806)(NOTES(I,MODE),I=1,4),NR1
C      806 FORMAT($1$,10X,4A4,2X,$NUMBER$,2X,1I1)
C      824 CONTINUE
C      CALL ZPL0T(Y,TEMPX,NPT,NPT)
C      IF(IWAY.NE.1) GO TO 3500
C      WRITE(6,8888)(I,LOGICS(I),I=1,9)
C      WRITE(6,8889)(I,LOGICS(I+9),I=1,9)
C      8888 FORMAT($0$,9($X,' ',$1I5,4X))

```



```
8889 FORMAT($ $,9($Y($,111,$)= $,115,4X))  
3500 CONTINUE  
C  
    RETURN  
    END
```

```
SUBROUTINE FOURZ(DATA,NN,NDIM,ISIGN)
```

```
C  
C THIS SUBROUTINE IS IDENTICAL TO THE COMPUTER LAB SUBROUTINE  
C FOUR2, AND IS, THEREFORE, NOT REPRODUCED HERE.
```



```

C
C SUBROUTINE FSAMP(NR1,NR2,IWAY,RAT,KD,NPT,KPART1,KPART2,Y,MODE)
C
C SUBROUTINE TO SAMPLE FUNCTIONS AND RETURN SAMPLES IN Y.
C ARGUMENTS ARE THE SAME AS THOSE OF SUBROUTINE FRESPEC
C EXCEPT Y, WHICH IS A VECTOR CONTAINING SAMPLES. 8DD
C NUMBERED ELEMENTS CONTAIN REAL PARTS OF SAMPLES. EVEN
C NUMBERED ELEMENTS CONTAIN IMAGINARY PARTS.
C
C
C INTEGER ASIG,BSIG,AMP,FREQ
C INTEGER BUTTN,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPOD
C INTEGER TEMPSL,TEMPBP
C COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
C COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
C COMMON KFRE,KAMP,KS2,KSIG,I8,I8TOP,IT19P
C COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
C * FREQ(72),KAAK,KDIV,KUP
C COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
C COMMON/KDS/KDISP(8,0:9,3)
C COMMON/IGRAF/I8X(8),IGRID(35),BUTTN(2),IFIG(100)
C COMMON IW8KA(10),IXH9LD(15),TEMPX(10),TEMPY(10),
C * TEMPOD(10),TEMPOD(10),TEMPSL(8),TEMPBP(8)
C COMMON KOUTIN(15),IW8KB(15)
C COMMON/INF0/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
C * IDC1,IDF1
C DOUBLE PRECISION SLP1,SLP2,YDEL1,YDEL2,DEL1,DEL2,
C * YVAL1,YVAL2,XRUN1,XRUN2,YT,BIG,AJUST,DX
C DIMENSION Y(1)
C XRUN1=XRUN2=-1000.0
C K1=K2=2
C
C IF(SENSESWITCH 1)205,206
C 205 OUTPUT(6)NR1,NR2,IWAY,RAT,KD,NPT,KPART1,KPART2,MODE
C PAUSE
C 206 CONTINUE

```



```

PART1=KPART1*2-1000.0
PART2=KPART2*2-1000.0

C.....UNPACK FIRST FIGURE
C
D0 10 I=1,8
10 CALL UNPACK(KDISP(I,NR1,MODE),TEMPX(I+1),TEMPY(I+1),IMD)
C
TEMPX(1)=-1000
TEMPY(1)=0
IF(MODE.EQ.2)TEMPY(1)=TEMPY(9)
D0 5 I=1,9
LOGICS(I)=(TEMPX(I)+1000)/2
5 LOGICS(I+9)=TEMPY(I)
C
C.....DETERMINE DELTA X
DEL1=DEL2=2*KPART1/(NPT-1.0)
IF(IWAY.NE.1) DEL1=2000.0/(NPT-1)
C
IF(IWAY.EQ.1) G0 T0 45
C
C.....UNPACK A.M. FUNCTION
C
D0 20 I=1,8
20 CALL UNPACK(KDISP(I,NR2,2),TEMPDX(I+1),TEMPDY(I+1),IMD)
TEMPDX(1)=-1000
TEMPDY(1)=TEMPDY(9)
C
C.....DETERMINE SLOPE
C
DX=TEMPDX(2)-TEMPDX(1)
SLP2=(TEMPDY(2)-TEMPDY(1))/DX
YDEL2=SLP2*DEL2
YVAL2=TEMPDY(1)
DEL1=DEL2*RAT

```



```

90 T0 100
92 CONTINUE
IF(XRUN2.LE.PART2) G0 T0 93
Y(K)=Y(K+1)=0.0
G0 T0 100
93 CONTINUE
Y(K)=YVAL1*YT
Y(K+1)=0
100 CONTINUE
RETURN
C
110 CONTINUE
KK=K1+1
YVAL1=TEMPY(K1)
IF(KK.EQ.10.AND.IWAY.EQ.1) G0 T0 60
IF(KK.LT.10.AND.XRUN1.LE.1000.0) G0 T0 111
KK=2
K1=1
K1=0
111 CONTINUE
DX=TEMPX(KK)-TEMPX(K1)
SLP1=(TEMPY(KK)-TEMPY(K1))/DX
YDEL1=SLP1*DEL1
K1=K1+1
G0 T0 60
120 CONTINUE
KK=K2+1
YVAL2=TEMPDY(K2)
IF(K2.EQ.10.0R.XRUN2.GE.1000.0) G0 T0 70
DX=TEMPDX(KK)-TEMPDX(K2)
SLP2=(TEMPDY(KK)-TEMPDY(K2))/DX
YDEL2=SLP2*DEL2
K2=K2+1
G0 T0 70
C
END

```



```

C
C SUBROUTINE DUTIN(I0B,NND,LI0,/LINE(NL)/)
C
C THIS SUBROUTINE DISPLAYS AND ERASES TEXT AND HANDLES USER
C INPUTS AT THE AGT TELETYPEWRITER
C
C I0B:-1= ERASE TEXT INDICATED
C      0=DISPLAY TEXT INDICATED
C      +1=DISPLAY TEXT INDICATED AND WAIT FOR AN INPUT. THEN
C CHECK INPUT AND RETURN STATUS IN NND.
C LI0=THE VECTORS OF TEXT MATRIX TXT TO BE DISPLAYED.
C
C INTEGER ASIG,BSIG,AMP,FREQ
C INTEGER BUTTN,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
C INTEGER TEMPSL,TEMPBP
C COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
C COMMON/LINDEF/ILIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
C COMMON KPRE,KAMP,KS2,KSIG,I8,IBT0P,ITT0P
C COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
C      * FREQ(72),KAAK,KDIV,KUP
C COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
C COMMON/KDS/KDISP(8,0:9,3)
C COMMON/IGRAF/IB9X(8),IGRID(35),BUTTN(2),IFIG(100)
C COMMON IWRKA(10),IXH8LD(15),TEMPX(10),TEMPY(10),
C      * TEMPDX(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
C COMMON KOUTIN(15),IWRKB(15)
C COMMON/INFO/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
C      * IDC1,IDF1
C
C.....NND=1:OK
C.....NND=2:ERROR
C.....NND=3:ALL BLANKS
C.....NND=4:C
C.....NND=5:END
C

```



```

C      IF(I08.EQ.-1) G0 T0 190
C      NND=1
C
C      904 F0RMAT($
C              $)
C
C      D0 100 I=1,NL
C      CALL TEXT0(IDEV,TXT(1,LINE(I)),12,ILIN(LINE(I)),25,1,3,IER)
C      100 C0NTINUE
C
C      IF(I08.EQ.0) RETURN
C
C      ENCODE(8,904,TXT(1,LI9))
C      CALL TEXTR(IDEV,TXT(1,LI9),2,ILIN(LI9),25,2,3,IER)
C      101 IF(M0D(ITDIR(LI9-5),8).EQ.0) G0 T0 101
C      CALL TEXTI(IDEV,K0UTIN,2,ILIN(LI9),25,IER)
C      DECODE(8,901,K0UTIN)NND
C      901 F0RMAT(1R3)
C      IF(NND.EQ.4H0END)NND=5
C      IF(NND.EQ.5) G0 T0 2000
C      150 C0NTINUE
C
C      DECODE(8,900,K0UTIN)(IWORKB(J),J=1,8)
C      900 F0RMAT(8R1)
C
C      D0 3000 I=1,8
C      IF(IWORKB(I).NE.48) G0 T0 3005
C      3000 C0NTINUE
C
C      NND=3
C      G0 T0 2000
C
C      3005 C0NTINUE
C      K0K=IWORKB(1)
C      IF(K0K.NE.19) G0 T0 3006

```



```

NND=4
GO TO 2000
3006 CONTINUE
IF((K9K.LE.9.AND.K9K.GE.0).OR.(K9K.EQ.16).OR.(K9K.EQ.32))
* GO TO 1001
GO TO 1002
1001 CONTINUE
NND=1
C
DO 1000 I=2,8
K9K=IWGRKB(I)
IF(K9K.EQ.59) GO TO 2000
IF(K9K.GT.9. OR.K9K.LT.0) GO TO 1002
1000 CONTINUE
GO TO 2000
1002 NND=2
2000 CONTINUE
IF(I93.EQ.1) RETURN
C
190 CONTINUE
DO 200 I=1,NL
DO 195 J=1,12
195 TXT(J,LINE(I))=4H
CALL TEXT9(IDEV,TXT(1,LINE(I)),12,ILIN(LINE(I)),25,1,1,IER)
200 CONTINUE
C
RETURN
END

```



```

C
C
C
C
C
SUBROUTINE ZPLOT(XY,JXY,N,NDIM,/ARG(NP1)/)

THIS IS A VERSION OF THE COMPUTER LAB SUBROUTINE VPL0T,
MODIFIED TO ALLOW EITHER A HISTOGRAM TYPE PLOT OR THE
NORMAL POINT PLOT.

DIMENSION IGRID(101),XS(11),YS(13),ICHAR(7),XY(1),JXY(1)
DATA ICHAR/1H+,1H*,1HΔ,1H$,1H=,1H.,1H /
DATA KOKD/1H1/
EQUIVALENCE(XS(1),LCUR),(XS(2),LSCALE)
NCUR=1
IXAX=50
IYAX=0
ISCALE=XL=XU=YL=YU=0
IF(NP1.LT.1) GO TO 2
XS(1)=ARG(1)
NCUR=LCUR
2 CONTINUE
IF(NP1.LT.6) GO TO 1234
XS(2)=ARG(2)
ISCALE=LSCALE
XL=ARG(3)
XU=ARG(4)
YL=ARG(5)
YU=ARG(6)
1234 CONTINUE
KUR=IABS(NCUR)
XS(1)=XL
XMAX=XU
YMIN=YL
YS(1)=YU
IF(ISCALE.NE.0) GO TO 32
XMAX=-1.0E 20
XS(1)=-XMAX
YS(1)=XMAX

```



```

YMIN=XS(1)
J2=0
D9 31 J=1,KUR
J2=J2+2
JIX=(JXY(J2-1)-1)*NDIM
JIY=(JXY(J2)-1)*NDIM
D9 31 I=1,N
IUX=JIX+I
IUY=JIY+I
IF(XY(IJX).GT.XMAX)XMAX=XY(IJX)
IF(XY(IJX).LT.XS(1))XS(1)=XY(IJX)
IF(XY(IJY).GT.YS(1))YS(1)=XY(IJY)
IF(XY(IJY).LT.YMIN)YMIN=XY(IJY)
31 CONTINUE
32 XR=XMAX-XS(1)
IF(XR.EQ.0.0)XR=1.0E-20
YR=YS(1)-YMIN
IF(YR.EQ.0.0)YR=1.0E-20
XT=XMAX*XS(1)
YT=YMIN*YS(1)
IF(XT.LT.0.0)IYAX=100.0*(-XS(1))/XR+1.5
IF(YT.LE.0.0)IXAX=48.0*YS(1)/YR+1.5
XMAX=XR/10.0
D9 46 I=2,11
46 XS(I)=XS(I-1)+XMAX
XMAX=YR/12.0
D9 47 I=2,13
47 YS(I)=YS(I-1)-XMAX
WRITE(6,10)(XS(I),I=1,11)
II=1
KK=0
D9 146 LINE=1,49
D9 101 J=1,101
101 IGRID(J)=ICHR(7)
IF(YT.GT.0.0) G9 T9 109

```



```

IF(LINE.NE.IXAX)G0 T0 109
D0 105 J=1,101
105 IGRID(J)=ICHAR(6)
109 IF(XT.LT.0.0) IGRID(IYAX)=ICHAR(6)
J2=0
D0 125 J=1,KUR
J2=J2+2
JIX=(JXY(J2-1)-1)*NDIM
JIY=(JXY(J2)-1)*NDIM
JC=M9D(J,5)
D0 125 I=1,N
IUX=JIX+I
IUY=JIY+I
IPTY=48.0*(YS(1)-XY(IJY))/YR+1.5
IF(IPTY.GT.49) IPTY=49
IF(IPTY.LT.1)IPTY=1
IF(LINE.EQ.IPTY.6R.(J.EQ.1.AND.NCUR.LT.0.AND.((LINE.GT.IPTY.AND.
* LINE.LT.IXAX).6R.(LINE.LT.IPTY.AND.LINE.GT.IXAX)))G0 T0 1313
G0 T0 125
1313 CONTINUE
IPTX=100.0*(XY(IJX)-XS(1))/XR+1.5
IF(IPTX.LT.1)IPTX=1
IF(IPTX.GT.101)IPTX=101
IF(JC.NE.0) G0 T0 119
IGRID(IPTX)=ICHAR(5)
G0 T0 125
119 IGRID(IPTX)=ICHAR(JC)
IF(NCUR.LT.0.AND.J.EQ.1) IGRID(IPTX)=KDKD
125 CONTINUE
IF(KK.GT.0) G0 T0 134
WRITE(6,20)YS(II),(IGRID(I),I=1,101),YS(II)
II=II+1
G0 T0 135
134 WRITE(6,30)(IGRID(I),I=1,101)
135 KK=KK+1

```



```

IF(KK.NE.4) GO TO 146
KK=0
146 CONTINUE
WRITE(6,40)(XS(I),I=1,11)
RETURN
20 FORMAT(1PE10.2,1H+,101A1,1H+,E9.2)
30 FORMAT(10X,1H*,101A1,1H*)
40 FORMAT(10X,1H*,20(5H+****),2H+*/1PE16.2,10E10.2)
10 FORMAT($ $,//$,$,1PE15.2,10E10.2/10X,1H*,20(5H+****),2H**)
END

```



```

C
C
C
C
C
C
SUBROUTINE FTINIT(ITAPE)
SUBROUTINE TO INITIALIZE DATA AND SYSTEM DURING SYSTEM START-UP.
ITAPE: 1=USER DEFINITIONS ON TAPE ARE TO BE TRANSFERRED TO DRUM.
        2=NO DATA IS ON TAPE.

INTEGER ASIG,BSIG,AMP,FREQ
INTEGER BUTTON,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
INTEGER TEMPSL,TEMPBP
COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
COMMON/LINDEF/LIN(6:10),ICHAN(20),LOGICT(10),LOGICS(32)
COMMON KFER,KAMP,KSG2,KSIG,I8,IBTOP,ITOP

COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
*   FREQ(72),KAAR,KDIV,KUP
COMMON KHL(3),TXT(12,6:10),ITDIR(50),IUDIR(10),NULL(12),IDEV
COMMON/KDS/KDISP(8,0:9,3)
COMMON/IGRAF/IBOX(8),IGRID(35),BUTTON(2),IFIG(100)
COMMON IWBRKA(10),IXHOLD(15),TEMPX(10),TEMPY(10),
*   TEMPDX(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
COMMON KROUTIN(15),IWBRKB(15)
COMMON/INF0/NOTES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
*   IDC1,IDF1

C
C
C
C
C
C
DIMENSION IWD(9)
DATA IWD/8,35,400,400,10,2,2,2,2/

CALL FIX
DB 291 I=1,72
291 FREQ(I)=0
I8=8
IBTOP=1
ITGP=1000
3000 CONTINUE

```



```

OUTPUT(102),'IF STORAGE TAPE NEW,TYPE NEW;IF OLD,TYPE OLD'
READ(101,800)ITAPE
800 FORMAT(1R3)
IF(ITAPE.EQ.4HNEW) GO TO 9030
IF(ITAPE.EQ.4HOLD) GO TO 9000
GO TO 3000
9000 ITAPE=1
GO TO 2500
9030 ITAPE=2
2500 CONTINUE
C
D9 1 I=1,12
1 NULL(I)=4H
KTN=77777777B
IPS=4HE000
EOM 030015
POT IPS
SKS 030400
BRU $-1
OUTPUT(102) 'READY ANALOG COMPUTER'
PAUSE 0000
C
C
WRITE(102,802)
802 FORMAT($ INPUT AGT NUMBER, 1 OR 2$)
92 CONTINUE
READ(101,801)IDEV
801 FORMAT(1R1)
IF(IDEV.NE.1.AND.IDEV.NE.2) GO TO 92
C
C
CALL DTINIT(IDEV,ITDIR,45,IER)
CALL DGINIT(IDEV,IGDIR,10,IER)
C
C

```



```

D8 100 I=6,10
CALL TEXT8(IDEV,NULL,12,ILIN(I),25,1,1,IER)
CONTINUE
100
C
CALL TEXT8(IDEV,NULL,12,25,1,1,1,IER)

D8 110 I=600,1000
ISIG(I)=0
CONTINUE
110
IRS=4H1000
E8M 030015
P8T IRS
SKS 030400
BRU $-1
C
C

D8 200 I=1,9
CALL GRAPH8(IDEV,ISIG(600),IWD(I),I,IER)
CONTINUE
200
C
KHED(1)=IHEAD(0,8)
LOGICT(1)=IHEAD(0,2)
KHED(2)=IPACK(-1000,0,0)
IB8X(1)=IHEAD(0,4)
IGRID(1)=IHEAD(1,3)
C
C

D8 1000 J=0,9
D8 1000 K=1,8
ASIG(K,J,1)=BSIG(K,J,1)=ICHAN(K)
ASIG(K,J,2)=ICHAN(9)
BSIG(K,J,2)=ICHAN(10)
ASIG(K,J,3)=ICHAN(12)
BSIG(K,J,3)=ICHAN(13)
CONTINUE
1000
S
E8M 030015

```



```

S      POT IPS
S      SKS 030400
S      BRU $-1
C
      DO 1010 I=1,72
      T16=819.2*2**((I-1)/12.0)
      FREQ(I)=T16*2**9+ICHAN(14)
      IF(I.GT.50)FREQ(I)=FREQ(50)
1010 CONTINUE
C
      DO 1020 I=0,9
      T16=I*1638*2**9
      AMP(I)=ICHAN(15)+T16
1020 CONTINUE
C
      IDC1=L8CF(IFIG(2))
      IDF1=L8CF(IFIG(52))
      IRR2=L8CF(TEMPX(2))
      RETURN
      END

```



```

C
C
C
C
SUBROUTINE TAPEDRUM(ITAPE)
SUBROUTINE T0 TRANSFER DATA T0, FROM, 0R BETWEEN DRUM,TAPE,
AND CORE.
INTEGER ASIG,BSIG,AMP,FREQ
INTEGER BUTTN,TXT,T16,T26,T36,TEMPX,TEMPY,TEMPDX,TEMPDY
INTEGER TEMPSL,TEMPBP
COMMON/FACS/BPFAC(3),SLFAC(3),SLMAX(3),SLMIN(3)
COMMON/LINDEF/ILIN(6:10),ICHAN(20),L9GICT(10),L9GICS(32)
COMMON KFRE,KAMP,KS2,KSIG,I8,I8TOP,ITT9P
COMMON ASIG(8,0:9,3),BSIG(8,0:9,3),ISIG(1000),AMP(0:9),
* FREQ(72),KAAK,KDIV,KUP
COMMON KHED(3),TXT(12,6:10),ITDIR(50),IGDIR(10),NULL(12),IDEV
COMMON/KDS/KDISP(8,0:9,3)
COMMON/IGRAF/IBGX(8),IGRID(35),BUTTN(2),IFIG(100)
COMMON IWBRKA(10),IXH9LD(15),TEMPX(10),TEMPY(10),
* TEMPDX(10),TEMPDY(10),TEMPSL(8),TEMPBP(8)
COMMON KOUTIN(15),IW0RKB(15)
COMMON/INF0/N0TES(4,40),IBIT(12),KALL,KTN,IRR1,IRR2,IRR3,IRR4,
* IDC1,IDF1
DIMENSION KT1(1562),KT2(240)
EQUIVALENCE(ASIG(1,0,1),KT1(1)),(KDISP(1,0,1),KT2(1))
C
C.....ITAPE=1.....TAPE T0 DRUM
C.....ITAPE=2.....WRITE 0N DRUM ONLY
C.....ITAPE=3.....DRUM T0 TAPE
IF(ITAPE.NE.2) REWIND 18
CALL E0FSET(LEND,IUNIT)
GO T0 (9000,9030,9050) ITAPE
9000 CONTINUE
ASSIGN 9025 T0 LEND
C
DO 9005 I=51,75
K=I

```



```

S S SKS 014000
S S BRU $-1
S S READ(18)(KT1(J),J=1,1562),(KT2(J),J=1,240),T16,IBT0P,ITT0P
S S SKS 014000
S S BRU $-1
S S REWIND I
S S SKS 014000
S S BRU $-1
S S WRITE(I)(KT1(J),J=1,1562),(KT2(J),J=1,240),T16,IBT0P,ITT0P
9005 CONTINUE
C
C RETURN
C
C 9025 CONTINUE
C
C DO 9010 I=K,75
C REWIND I
S S SKS 014000
S S BRU $-1
S S WRITE(I)(KT1(J),J=1,1562),(KT2(J),J=1,240),T16,IBT0P,ITT0P
9010 CONTINUE
C
C RETURN
C
C 9030 CONTINUE
C K=51
S S GO TO 9025
9050 CONTINUE
C REWIND 18
C
C DO 9075 I=51,75
C REWIND I
S S SKS 014000
S S BRU $-1
S S READ(I)(KT1(J),J=1,1562),(KT2(J),J=1,240),T16,IBT0P,ITT0P

```



```

S      SKS 014000
S      BRU $-1
      WRITE(18)(KT1(J),J=1,1562),(KT2(J),J=1,240),T16,IBT0P,ITT0P
9075   CONTINUE
      ENDFILE 18
      REWIND 18
      RETURN
      END

```

```

$FIX  PZE      0
      BRM      R\RSTS
      PZE      1
      PZE      FILE
      LDA      3,1
      EOR      =020000000
      STA      3,1
      MP0      FIX
      BRR      FIX
      TEXT     8,18
      PZE      2
      END

```


LIST OF REFERENCES

1. Adage Inc., System Reference Manual for Adage Graphics Terminal.
2. University of Utah Technical Report UTEC-CSc-71-117, Electronics, Music and Computers, by A.C. Ashoton, December 1971.
3. Beardslee, D.C. and Wertheimer, M., Readings in Perception, p. V, Van Nostrand Company, 1964.
4. Bekey, G.A. and Karplus, W.J., Hybrid Computation, John Wiley & Sons, 1968.
5. Beranek, L. L., "Digital Synthesis of Speech and Music," IEEE Transactions on Audio and Electroacoustics, Vol. AU-18, No. 4, p. 426-433, December 1970.
6. Bertram, S., "Frequency Analysis Using the Discrete Fourier Transform," IEEE Transactions on Audio and Electroacoustics, Vol. AU-18, No. 4, p. 495-500, December 1970.
7. Carrington, J. F., "The Talking Drums of Africa," Scientific American, p. 104-113, December 1971.
8. Clapper, G. L., "Automatic Word Recognition," IEEE Spectrum, p. 57-69, August 1971.
9. Comcor/Astrodata, Inc., Operation and Instruction Manual for USN Monterey CI-5000.
10. Cooley, J.W., Lewis, P.A., and Welch, P.D., "Application of the Fast Fourier Transform to Computation of Fourier Integrals, Fourier Series, and Convolution Integrals," IEEE Transactions on Audio and Electroacoustics, Vol. AU-15, No. 2, p. 79-84, June 1967.
11. Culver, C. A., Musical Acoustics, McGraw-Hill, 1956.
12. Dorrian, L. V., Digital Spectral Analysis, Masters Thesis, United States Naval Postgraduate School, 1968.
13. Flanagan, J. L., and others, "Synthetic Voices for Computers," IEEE Spectrum, Vol. 7, No. 10, p. 22-45, October 1970.
14. Fletcher, H., Speech and Hearing in Communications, p. V, Van Nostrand Company, 1953.

15. Focht, L.R., "The Single Equivalent Formant," 1966
IEEE International Communications Conference Digest of
Technical Papers, Vol. 2, p. 108, Lewis Winner, 1966.
16. Freedman, M.D., "Analysis of Musical Instrument Tones,"
Journal of Acoustical Society of America, Vol. 41,
No. 5, p. 1265-1271, 1967.
17. Glisson, T.H., Black, C.I., and Sage, A.P., "The Digital
Computation of Discrete Spectra Using the Fast Fourier
Transform," IEEE Transactions on Audio and Electro-
acoustics, Vol. AU-18, No. 3, p. 271-287, September 1970.
18. Green, D.M., "Psychoacoustics and Detection Theory,"
Journal of Acoustical Society of America, No. 32,
p. 1189-1203, 1960.
19. M.I.T. Research Laboratory of Electronics Technical
Report 484, Perception of Musical Intervals: Evidence
for the Central Origin of the Pitch of Complex Tones,
by Andrianus J.M. Houtsma and Julius L. Goldstein,
October 1, 1971.
20. Ito, M.R. and Donaldson, R.W., "Zero-Crossing Measure-
ments for Analysis and Recognition of Speech Sounds,"
IEEE Transactions on Audio and Electroacoustics,
Vol. AU-19, No. 3, p. 235-242, September 1971.
21. Maissis, A., Walrave, P.H., and Fievet, F., "A New
Analog-Digital Filtering Method for Real-Time Speech
Recognition," IEEE Transactions on Audio and Electro-
acoustics, Vol. AU-18, No. 4, p. 385-388, December 1970.
22. Moorer, J.A., "Music and Computer Composition,"
Communications of the ACM, Vol. 15, No. 2, p. 104-113,
February 1972.
23. Potter, R.K., Kopp, G.A., and Kopp, H.G., Visible
Speech, Dover, 1966.
24. Risset, J.C. and David, E.E., "Analysis of Musical
Instrument Tones," Physics Today, Vol. 22, No. 2,
p. 23-30, February 1969.
25. Schaffer, R.W., "A Survey of Digital Speech Processing
Techniques," IEEE Transactions on Audio and Electro-
acoustics, Vol. AU-20, No. 1, p. 28-35, March 1972.
26. Slaymaker, F.H., "Chords from Tones Having Stretched
Partials," Journal of Acoustical Society of America,
Vol. 47, No. 6, Part 2, p. 1569-1571, 1970.

27. Teacher, C.F., Kellet, H.G., and Focht, L.F.,
"Experimental, Limited Vocabulary, Speech Recognizer,"
IEEE Transactions on Audio and Electroacoustics,
Vol. AU-15, No. 3, p. 127-130, September 1967.
28. Tove, P.A., Ejdesjo, L., and Svardstrom, A., "Frequency
and Time Analysis of Polyphonic Music," Journal of
Acoustical Society of America, Vol. 41, No. 5, p. 1265-
1271, 1967.
29. Xerox Data Systems, XDS FORTRAN IV Reference Manual,
Pub. No. 90 11 07, 1968.
30. Xerox Data Systems, XDS Real-Time Monitor Reference
Manual, Pub. No. 90 11 08E, 1969.
31. Xerox Data Systems, Hybrid Computer System Technical
Manual, Pub. No. 98 03 61A, 1969.
32. Xerox Data Systems, 9300 Computer Reference Manual,
Pub. No. 90 00 50, 1967.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Asst. Prof. V. Michael Powers Electrical Engineering Department Naval Postgraduate School Monterey, California 93940	1
4. LT. James William Mizerski 555 N. Alhambra Road San Gabriel, California 91775	1
5. Computer Laboratory Electrical Engineering Department Naval Postgraduate School Monterey, California 93940	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)		2a. REPORT SECURITY CLASSIFICATION	
Naval Postgraduate School Monterey, California 93940		Unclassified	
REPORT TITLE		2b. GROUP	
An Interactive Hybrid Computer System for Time Domain Audio Synthesis			
DESCRIPTIVE NOTES (Type of report and, inclusive dates)			
Master's Thesis: March 1973			
AUTHOR(S) (First name, middle initial, last name)			
James William Mizerski			
REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS	
March 1973	200	32	
CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)		
PROJECT NO.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)		
DISTRIBUTION STATEMENT			
Approved for public release; distribution unlimited.			
SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
		Naval Postgraduate School Monterey, California 93940	
ABSTRACT			
<p>A time domain oriented technique using periodic functions composed of first order segments was employed in the development of a real-time interactive computer audio synthesis system. A general purpose analog computer was programmed to create non-monotonic function generators which are used to generate sequences of periodic functions that can be frequency and amplitude modulated. The system is appropriate for experiments in psychoacoustics. Exploration of the relationships between the physical and the perceptual significance of synthesis parameters of the technique is aided by real-time frequency analysis programs and an audio output capability. The system includes an interactive graphics terminal for conversational I/O and conventional peripheral devices for creating permanent records. In preliminary testing, the technique and system displayed promising possibilities. Modifications to explore specific areas of perception in greater depth were considered.</p>			

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
computers						
hybrid computation						
audio synthesis						
psychoacoustics						
electroacoustics						
function generators						
computer graphics						
time domain synthesis						



143342

Thesis
M654
c.1

Mizerski

An interactive hybrid
computer system for time
domain audio synthesis.

2

d
ne

143342

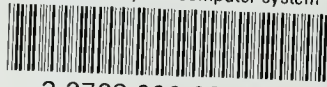
Thesis
M654
c.1

Mizerski

An interactive hybrid
computer system for time
domain audio synthesis.

thesM654

An interactive hybrid computer system fo



3 2768 000 98475 1

DUDLEY KNOX LIBRARY





3 2768 000 98475 1
DUDLEY KNOX LIBRARY